(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 4 120 065 A1**

(12) **EUROPEAN PATENT APPLICATION**
published in accordance with Art. 153(4) EPC

(43) Date of publication:
18.01.2023 Bulletin 2023/03

(21) Application number: **21845550.9**

(22) Date of filing: **15.06.2021**

(51) International Patent Classification (IPC):
G06F 8/41 *(2018.01)*

(52) Cooperative Patent Classification (CPC):
A63F 13/52; G06T 1/20; G06F 8/41

(86) International application number:
PCT/CN2021/100026

(87) International publication number:
WO 2022/017070 (27.01.2022 Gazette 2022/04)

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
KH MA MD TN

(30) Priority: **21.07.2020 CN 202010704005**

(71) Applicant: **Tencent Technology (Shenzhen) Company Limited**
Shenzhen, Guangdong 518057 (CN)

(72) Inventors:
- **ZHAO, Xinda**
  Shenzhen, Guangdong 518057 (CN)
- **GONG, Zhipeng**
  Shenzhen, Guangdong 518057 (CN)

(74) Representative: **Gunzelmann, Rainer et al**
Wuesthoff & Wuesthoff
Patentanwälte PartG mbB
Schweigerstraße 2
81541 München (DE)

(54) IMAGE PROCESSING METHOD AND APPARATUS, AND SERVER AND MEDIUM

(57) An image processing method and apparatus, and a server and a medium. The method comprises: in response to an operation event regarding a target game running in a target container, acquiring a target shading code corresponding to the operation event (S201); acquiring, from game resources, a target compilation result of the target shading code, wherein the game resources comprise at least one of the following: a game mirror image of the target game, and a target shared directory between the target container and an associated container of the target container, and the target shared directory comprises a first compilation result obtained by means of the associated container compiling at least one shading code during the game running process (S202); and performing graphic processing according to the target compilation result to obtain a feedback image corresponding to the operation event (S203).

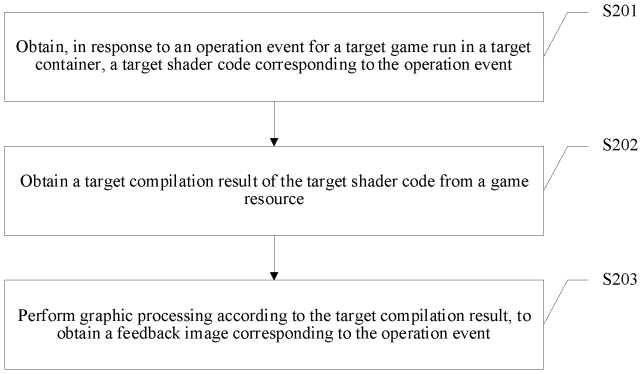

FIG. 2

EP 4 120 065 A1

Description

RELATED APPLICATION

[0001] This application claims priority to Chinese Patent Application No. 202010704005.8, entitled "IMAGE PROCESSING METHOD AND APPARATUS, SERVER, AND MEDIUM" and filed with the China National Intellectual Property Administration on July 21, 2020, which is incorporated herein by reference in its entirety.

FIELD OF THE TECHNOLOGY

[0002] This application relates to the field of Internet technologies, specifically, to the field of image processing technologies, and in particular, to an image processing method, an image processing apparatus, a server, and a computer storage medium.

BACKGROUND OF THE DISCLOSURE

[0003] With the development of Internet technologies, cloud gaming has attracted much attention. The cloud gaming may also be referred to as gaming on demand, and is a gaming mode based on cloud computing. The cloud computing is an Internet-based computing mode. Cloud gaming technology allows a client device with relatively limited graphic processing and data computing capabilities to run a high-quality game. In a cloud gaming scenario, a game is run on a cloud server instead of a game client of a player. The cloud server renders a game scene into video and audio streams, and transmits the video and audio streams to the game client of the player through a network for playback. The game client of the player does not need to have high graphic processing and data computing capabilities, and only needs to have a basic streaming playback capability and a capability of obtaining an input instruction of the player and sending the input instruction to the cloud server.

[0004] For a cloud game, a duration consumed during game running is an important indicator for measuring a running speed of the cloud game. When the duration consumed during game running is longer, the running speed of the cloud game is slower. When the duration consumed during game running is shorter, the running speed of the cloud game is faster. In view of this, how to shorten the duration consumed during game running to increase the running speed of the cloud game has become a research hotspot.

SUMMARY

[0005] Embodiments of this application provide an image processing method and apparatus, a server, and a medium, to effectively shorten a duration of compiling a target shader code, thereby effectively increasing the running speed of a target game.

[0006] According to an aspect, an embodiment of this application provides an image processing method, including:

obtaining, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event;

obtaining a target compilation result of the target shader code from a game resource, the game resource including at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared directory including a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and

performing graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event.

[0007] According to another aspect, an embodiment of this application provides an image processing apparatus, including:

an obtaining unit, configured to obtain, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event,

the obtaining unit being further configured to obtain a target compilation result of the target shader code from a game resource, the game resource including at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared directory including a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and

a processing unit, configured to perform graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event.

[0008] According to still another aspect, an embodiment of this application provides a server including an input interface and an output interface. The server further includes:

a computer storage medium, configured to store one or more instructions; and
a processor, configured to load the one or more instructions stored in the computer storage medium, to perform the following steps:

obtaining, in response to an operation event of

a target game running in a target container, a target shader code corresponding to the operation event;

obtaining a target compilation result of the target shader code from a game resource, the game resource including at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared directory including a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and

performing graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event.

[0009] According to yet another aspect, an embodiment of this application provides a computer storage medium, storing one or more instructions, the one or more instructions being configured to be loaded by a processor to perform the following steps:

obtaining, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event;

obtaining a target compilation result of the target shader code from a game resource, the game resource including at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared directory including a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and

performing graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] To describe the technical solutions in the embodiments of this application more clearly, the following briefly introduces the accompanying drawings required for describing the embodiments. Apparently, the accompanying drawings in the following description show only some embodiments of this application, and a person of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

FIG. 1a is a diagram of a system architecture of a cloud gaming system according to an embodiment of this application.

FIG. 1b is a schematic diagram of a connection between an edge server and a plurality of game clients according to an embodiment of this application.

FIG. 1c is a schematic diagram of transmitting a game picture to a game client for display according to an embodiment of this application.

FIG. 2 is a schematic flowchart of an image processing method according to an embodiment of this application.

FIG. 3a is a schematic diagram of producing a game mirror according to an embodiment of this application.

FIG. 3b is a schematic diagram of mounting a target shared directory according to an embodiment of this application.

FIG. 3c is another schematic diagram of mounting a target shared directory according to an embodiment of this application.

FIG. 3d is still another schematic diagram of mounting a target shared directory according to an embodiment of this application.

FIG. 3e is yet another schematic diagram of mounting a target shared directory according to an embodiment of this application.

FIG. 4 is a schematic flowchart of another image processing method according to another embodiment of this application.

FIG. 5a is a schematic diagram of mounting a target shared directory according to another embodiment of this application.

FIG. 5b is another schematic diagram of mounting a target shared directory according to another embodiment of this application.

FIG. 5c is still another schematic diagram of mounting a target shared directory according to another embodiment of this application.

FIG. 5d is yet another schematic diagram of mounting a target shared directory according to another embodiment of this application.

FIG. 5e is a schematic diagram of caching a target compilation result according to an embodiment of this application.

FIG. 6 is a schematic structural diagram of an image processing apparatus according to an embodiment

of this application.

FIG. 7 is a schematic structural diagram of a server according to an embodiment of this application.

DESCRIPTION OF EMBODIMENTS

[0011] Technical solutions in embodiments of this application are clearly and completely described below with reference to the accompanying drawings in the embodiments of this application.

[0012] To better run a cloud game, an embodiment of this application provides a cloud gaming environment. In the cloud gaming environment, a plurality of operating systems may be run on an independent server (for example, a server with an architecture such as an ARM architecture or an x86 architecture) by running a system container, and related images are transmitted to a remote receiving program through a video stream for processing. The ARM architecture is a processor architecture of a 32-bit or 64-bit reduced instruction set, and the x86 architecture is a computer language instruction set executed by a microprocessor. The container refers to a virtualized type on an operating system level and may be configured to carry an operating system. The container may be implemented by using an isolation mechanism (for example, a namespace). In a kernel mode, a plurality of operating systems (that is, a server operating system and a device operating system) share a same kernel; and in a user mode, the plurality of operating systems remain independent of each other. The server operating system refers to a general-purpose operating system, for example, a Linux operating system in the server. A host on which the server operating system resides may be referred to as a server host. The device operating system refers to an operating system, for example, an Android operating system, an IOS operating system, or the like in the container.

[0013] Correspondingly, the system container refers to an exemplary container and may be run based on the server operating system (for example, the Linux operating system). For example, the system container may be an Android container running on an open-source Linux operating system, and a plurality of Android containers may simultaneously run on one Linux operating system. An Android game mirror is loaded by the Android container. The Android game mirror refers to a mirror file including both basic Android-related files and game-related files. The mirror refers to a file storage form in which a plurality of files are combined into a single mirror file by using the mirror to facilitate distribution and use of the files. It is to be understood that the system container mentioned in this embodiment of this application is not limited to the Android container. For example, the system container may also be an IOS container when the IOS operating system supports open-source research and development. Based on the above, in the cloud gaming environment provided in this embodiment of this application, a large quantity of system containers may be deployed on an independent server, and powerful capabilities of a central processing unit (CPU) and a graphics processing unit (GPU) on a server side may be fully used, to implement highly concurrent execution of system operations, thereby increasing the running speed of the cloud game.

[0014] Based on the cloud gaming environment mentioned above, an embodiment of this application provides a cloud gaming system. Referring to FIG. 1a, the cloud gaming system may include at least one edge server 11 and a plurality of game clients 12. The edge server 11 refers to a server configured to run the system container. The server may be any independent physical server, or may be a cloud server that provides basic cloud computing services such as a cloud service, a cloud database, cloud computing, a cloud function, cloud storage, a network service, cloud communication, a middleware service, a domain name service, a security service, a content delivery network (CDN), big data, and an AI platform. As shown in FIG. 1b, at least one system container may be deployed inside each edge server 11, and each system container may be connected to at least one game client 12. The each system container may be configured to run one or more games, and in a process of running any game, the each system container may transmit a game picture of the any game to the game client 12 to which the system container is connected for display. The game client 12 may be any terminal device (which is referred to as a terminal for short) having basic capabilities such as a streaming playback capability, a man-machine interaction capability, and a communication capability, for example, a smartphone, a tablet computer, a notebook computer, a desktop computer, a smart television, or the like. Alternatively, the game client 12 may be an application (APP) running on the terminal device. It is to be understood that, FIG. 1a exemplarily shows a system architecture of a cloud gaming system and is not intended to limit a specific architecture of the cloud gaming system. For example, in other embodiments, the cloud gaming system may further include a mirror server configured to produce the game mirror, and the like.

[0015] One or more cloud games may be run based on the foregoing cloud gaming system. A running principle of any cloud game is as follows: after a game mirror of a cloud game is produced, the game mirror of the cloud game may be deployed inside an edge server, so that the edge server may enable a corresponding system container by loading the game mirror. When receiving a connection request related to the cloud game sent by a game client, the edge server may allocate a corresponding system container to establish a connection to the game client. After the allocated system container and the game client are successfully connected, the allocated system container may run the cloud game and render a game picture of the cloud game in real time, and then transmit the rendered game picture to the game client through a video stream for display, as shown in FIG. 1c. In a process

of displaying the game picture, the game client may send an operation event of a player for the game picture to the corresponding system container through a data stream. Correspondingly, the system container may be responsible for capturing and rendering a render buffer zone image (that is, a feedback image corresponding to the operation event) corresponding to the operation event, and returning the feedback image corresponding to the operation event to the game client for display.

[0016] When rendering the feedback image corresponding to the operation event, the system container usually involves image processing in several stages such as image shading, texture processing, and deformation. In view of this, an embodiment of this application provides a shared GPU shader cache mechanism for the image shading, to shorten a duration of performing the image shading, so that a rendering duration required for rendering the feedback image is shortened, and a total duration consumed during game running is shortened, thereby increasing the running speed of the cloud game. A GPU shader is a part of GPU hardware functions. When implemented by using software, a driver of the GPU may compile a GPU shader code written by a user into a target code executable by the GPU. A principle of a GPU shader cache mechanism is as follows: when a single system container caches a compilation result obtained by compiling the GPU shader code by the single system container to continuously run or rerun the cloud game in the single system container, if a same GPU shader code needs to be compiled by the system container, a compilation process may be skipped and the compilation result is directly used. Correspondingly, a principle of the shared GPU shader cache mechanism is as follows: a compilation result of a GPU shader code cached by a single system container is shared to other system containers, so that when a same GPU shader code needs to be compiled by the other system containers, a compilation process may be skipped and the shared compilation result is directly used.

[0017] In a specific implementation, the shared GPU shader cache mechanism provided in this embodiment of this application may mainly include the following two mechanisms:

(1) Static shared GPU shader cache mechanism

The static shared GPU shader cache mechanism refers to a shared mechanism in which a second compilation result of at least one GPU shader code is added to a game mirror, so that system containers loading the game mirror may all share the second compilation result. Specifically, a principle of the static shared GPU shader cache mechanism is generally as follows: the cloud game may be pre-run in a process of producing the game mirror of the cloud game; and in a process of running the cloud game, at least one GPU shader code is compiled to obtain the second compilation result, and then the second compilation result is added to the game mirror of the cloud game. In this way, when the game mirror of the cloud game is deployed inside system containers in different edge servers or different system containers in a same edge server, system containers loading the game mirror may all include the second compilation result. Further, when the system containers need to compile a GPU shader code, if the game mirror includes the second compilation results of the GPU shader code, a corresponding second compilation result may be directly obtained from the game mirror to perform subsequent processing, thereby increasing the running speed of a game. Based on the above, in the static shared GPU shader cache mechanism, since second compilation results of the GPU shader code are included in the game mirror, a problem of compilation result sharing between some identical games may be resolved without mounting an additional shared directory to the system container.

(2) Dynamic shared GPU shader cache mechanism

[0018] The dynamic shared GPU shader cache mechanism refers to a shared mechanism in which in the process of running the game, a same compilation result of the GPU shader code is shared between a plurality of system containers by using a shared directory. In a specific implementation, a principle of the dynamic shared GPU shader cache mechanism is generally as follows: in a process of running system containers, a shared directory may be mounted to the system containers as a GPU shader compilation cache directory, so that a first compilation result of each GPU shader code compiled by each system container in the process of running the game may be synchronously stored in the shared directory. In this way, when a GPU shader code needs to be compiled by a system container a, if the GPU shader code has been compiled by other system containers and a corresponding first compilation result has been cached in a target shared directory, the system container a may directly obtain the first compilation result from the target shared directory to perform subsequent processing, thereby increasing the running speed of the game.

[0019] In the dynamic shared GPU shader cache mechanism, since the shared directory may be mounted to a plurality of system containers, a compilation result of a GPU shader code newly generated by a single system container in a process of running a current game is cached in the shared directory, so that the compilation result may be dynamically shared and used by the plurality of system containers in real time. The first compilation result of the GPU shader code compiled by the single system container in the process of running the game is effectively used, to increase the running speed of a same game or different cloud games in other system containers. In view of this, the dynamic shared GPU shader cache mechanism provided in this embodiment of this application can resolve a problem of compilation

result sharing between some identical games and resolve a problem of GPU shader cache sharing between some different games.

[0020] It can be learned from the foregoing description that, both the static shared GPU shader cache mechanism and the dynamic shared GPU shader cache mechanism provided in this embodiment of this application can resolve the problem of compilation result sharing between the system containers to some extent. In addition, the cached compilation result is used to directly perform subsequent processing without performing the compilation process of the GPU shader code, so that a code compilation duration can be effectively shortened, thereby increasing the running speed of the cloud game. It is to be understood that, in an actual application, two shared GPU shader cache mechanisms provided in this embodiment of this application can be used alone or used in combination. This is not limited in this embodiment of this application.

[0021] Based on the related description of the two shared GPU shader cache mechanisms, an embodiment of this application provides an image processing method. The image processing method may be performed by the target container running in the target edge server in the cloud gaming system mentioned above. The target edge server is any edge server in the cloud gaming system, and the target container is any system container running based on a server operating system of the target edge server. Referring to FIG. 2, the image processing method may include the following steps S201 to S203.

[0022] S201: Obtain, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event.

[0023] The operation event is detected in a game picture of the target game displayed on a target game client that is connected to the target container. In a specific implementation, in a process of running the target game, the target container may send the game picture of the target game to the target game client through a video stream for display. Correspondingly, in a process of displaying the game picture, the target game client may detect whether there is an operation event of the target game. The operation event may include, but is not limited to: an event inputted by operating a physical control component (such as a mouse or a keyboard), an event inputted by controlling movement of a gravity sensing apparatus (such as a gyroscope), and an event inputted by touching any picture content on the game picture with a finger. After detecting the operation event, the target game client may transmit the operation event to the target container through a data stream. Correspondingly, after receiving the operation event, the target container may obtain a target shader code corresponding to the operation event.

[0024] S202: Obtain a target compilation result of the target shader code from a game resource.

[0025] In this embodiment of this application, the game resource may include at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container. The game mirror of the target game may be obtained based on the static shared GPU shader cache mechanism mentioned above. The game mirror of the target game may include a second compilation result of at least one shader code (that is, a GPU shader code) compiled by pre-running the target game, and each second compilation result is added to the game mirror in a process of producing the game mirror. The target shared directory between the target container and the associated container may be obtained based on the dynamic shared GPU shader cache mechanism mentioned above. The target shared directory may include a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container. It is to be understood that, the target shared directory is not limited to a first compilation result of the associated container, and may further include the first compilation result obtained by compiling, when running the game, at least one shader code by the target container. The associated container refers to a system container running an associated game of the target game. The associated game may include: the target game or a game using a same game engine as the target game.

[0026] It can be learned from the foregoing description that, in this embodiment of this application, the static shared GPU shader cache mechanism and the dynamic shared GPU shader cache mechanism mentioned above may be used alone or used in combination. The game resource may only include the game mirror of the target game when the static shared GPU shader cache mechanism is used alone. In this case, a specific implementation of step S202 may be that: whenthe game mirror of the target game includes a second compilation result of the target shader code, the second compilation result of the target shader code is directly obtained from the game mirror of the target game as the target compilation result; and when the game mirror of the target game does not include the second compilation result of the target shader code, the target shader code is compiled to obtain the target compilation result.

[0027] The game resource may only include the target shared directory between the target container and the associated container when the dynamic shared GPU shader cache mechanism is used alone. In this case, a specific implementation of step S202 may be that: when the target shared directory includes a first compilation result of the target shader code, directly obtaining the first compilation result of the target shader code from the target shared directory as the target compilation result; and when the game mirror of the target game does not include the first compilation result of the target shader code, compiling the target shader code to obtain the target compilation result. The first compilation result of the target shader code may be obtained by compiling the target shader code by the associated container in a proc-

ess of running the associated game; or may be obtained by compiling the target shader code by the target container in a historical process of running the target game. This is not limited in this embodiment of this application.

[0028] The game resource may include the game mirror of the target game and the target shared directory when the static shared GPU shader cache mechanism and the dynamic shared GPU shader cache mechanism are used in combination. In this case, a specific implementation of step S202 may be that: first detecting whether a second compilation result of the target shader code exists in the game mirror of the target game; obtaining the second compilation result of the target shader code from the game mirror as the target compilation result when the second compilation result exists; and obtaining the first compilation result from the target shared directory between the target container and the associated container of the target container as the target compilation result of the target shader code whenthe second compilation result does not exist. Whenthe target compilation result fails to be obtained from the target shared directory, it indicates that the target compilation result does not exist in the target shared directory, that is, the target shader code has not been compiled by the target container and the associated container. In this case, the target container may compile the target shader code to obtain the target compilation result.

[0029] S203: Perform graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event.

[0030] After step S203 is performed, the image processing method may further include: displaying the feedback image on the target game client.

[0031] After obtaining the target compilation result, the target container may perform a series of graphic processing according to the target compilation result by using the GPU, to obtain the feedback image corresponding to the operation event. The graphic processing may include, but is not limited to: alpha channel combining, and adjustment of a pixel value coordinate. After obtaining the feedback image, the target container may send the feedback image to the target game client through a video stream, to enable the target game client to display the feedback image.

[0032] In this embodiment of this application, after a target shader code corresponding to the operation event is obtained in response to an operation event of a target game running in a target container, a compilation process of the target shader code may be skipped to directly obtain a target compilation result of the target shader code from a game mirror or a target shared directory between the target container and an associated container of the target container. Then, graphic processing may be performed according to the target compilation result, to obtain a feedback image corresponding to the operation event; and the feedback image is displayed on a target game client. It can be seen that, in this embodiment of this application, graphic processing is performed by directly using the target compilation result compiled from the game mirror or the associated container in the target shared directory, so that a duration of compiling the target shader code can be effectively shortened, thereby increasing the running speed of a target game.

[0033] Based on the related description of the embodiments of the image processing method shown in FIG. 2, a specific process of producing the game mirror of the target game and a specific process of mounting the target shared directory involved in the method embodiment shown in FIG. 2 are respectively described as follows.

(1) Produce the game mirror of the target game based on a static shared GPU shader cache mechanism

Referring to FIG. 3 a, in a process of producing the game mirror of the target game, a basic mirror (such as an Android basic mirror) of a device operating system may be first loaded; then the target game may be installed to trigger compilation of a GPU shader code of a first portion related to the target game, to obtain a compilation result of the first portion; and the compilation result of the first portion is added to the game mirror of the target game. In an implementation, the compilation result of the first portion and the GPU shader code of the first portion may be directly added to the game mirror of the target game correspondingly. In another implementation, the compilation result of the first portion and an index value of the GPU shader code of the first portion may be added to the game mirror of the target game correspondingly. The index value may be obtained by performing a hash operation on the GPU shader code. Then, the target game may be run and an analog operation event of the target game may be inputted to trigger compilation of a GPU shader code of a second portion corresponding to the analog operation event, to obtain a compilation result of the second portion. Further, the compilation result of the second portion is added to the game mirror of the target game. In an implementation, the compilation result of the second portion and the GPU shader code of the second portion may be directly added to the game mirror of the target game correspondingly. In another implementation, the compilation result of the second portion and an index value of the GPU shader code of the second portion may be added to the game mirror of the target game correspondingly.

(2) Mount the target shared directory based on the dynamic shared GPU shader cache mechanism

[0034] First, the target edge server may select the target shared directory. Then, the target shared directory may be mounted to the target container and the associated container, so that in a process of running games in the target container and the associated container, a first compilation result of each compiled GPU shader code

may be synchronously cached in the target shared directory. Specifically, the target edge server may generate a directory mounting instruction carrying the target shared directory; and respectively send the directory mounting instruction to the target container and the associated container, so that the target container may mount the target shared directory to a dynamic cache directory in the target container, and the associated container may mount the target shared directory to a preset directory in the associated container. After the target shared directory is successfully mounted to the target container and the associated container, when the associated container has compiled at least one GPU shader code and obtained a corresponding first compilation result during running, the first compilation result obtained by compiling the GPU shader code by the associated container may be added to the target shared directory. Similarly, when the target container has compiled at least one GPU shader code and obtained a corresponding first compilation result during running, the first compilation result obtained by compiling the GPU shader code by the target container may be added to the target shared directory.

[0035] The target shared directory may support two selection manners.

[0036] In a first selection manner, a target shared directory (for example, /home/shadercache) on a server host (that is, a host) on which the server operating system of the target edge server resides is selected for use by all system containers running on the target edge server. In this selection manner, the target shared directory is located in the server host on which the server operating system resides. In an actual application, since the system container may generate a first compilation result of a corresponding GPU shader code in a process of running different types of cloud games, a data volume of first compilation results of GPU shader codes accumulated by a plurality of system containers is usually relatively large. When the plurality of system containers are mounted to a same shared directory, a data volume of compilation results cached in the shared directory is relatively large, and consequently it takes a relatively long time consumed to search the shared directory for a compilation result of a related GPU shader code. In view of this, the server host may determine, according to a quantity of types of the cloud games run on the system containers in the target edge server, a quantity of shared directories to be provided.

[0037] When the quantity of types of the cloud games run on each system container in the target edge server is relatively small (that is, the quantity of types of the cloud games is less than a quantity threshold), the server host may provide only one shared directory, so that the each system container in the target edge server may be mounted to the shared directory. In this case, the target edge server may use the shared directory provided by the server host as the target shared directory between the target container and the associated container. That is, in this implementation, FIG. 3b is a schematic diagram of mounting a target shared directory by a target container. As shown in FIG. 3b, since the server host only includes the target shared directory, the target shared directory may be used for mounting the target container and the associated container of the target container and also used for mounting system containers in the target edge server that are not related to the target container.

[0038] When the quantity of types of the cloud games run on the system containers in the target edge server is relatively large (that is, the quantity of types of the cloud games is greater than or equal to the quantity threshold), the server host may provide a plurality of shared directories. In this case, for related system containers running a same cloud game or related system containers of a cloud game sharing a same game engine, the related system containers are more likely to use a same GPU shader code, and therefore the related system containers may be mounted to a same shared directory. In addition, for unrelated system containers, the unrelated system containers may be mounted to other shared directories on the server host. In this case, the target edge server may randomly select a shared directory from the plurality of shared directories provided by the server host as the target shared directory between the target container and the associated container. That is, in this implementation, FIG. 3c is a schematic diagram of mounting a target shared directory by a target container. As shown in FIG. 3c, the server host may include the target shared directory and also include other shared directories other than the target shared directory. Therefore, the target shared directory may only be used for mounting the target container and the associated container of the target container, and the other shared directories in the server host may be used for mounting the system containers in the target edge server that are not related to the target container. It is to be understood that, when there are a large quantity of cloud games, the server host may still provide only one shared directory if an indicator of the time consumed to search the target shared directory for a compilation result of a related GPU shader code is ignored.

[0039] In a second selection manner, a target shared directory (for example, 192.168.10.10:/mnt/share) is selected from a network file system on the Internet for use by system containers in all edge servers on the Internet. The network file system is independent of each edge server in the cloud gaming system and allows access by a system container of the each edge server. In this selection manner, the target shared directory is located in the network file system. In a specific implementation, similar to the reason why the server host determines, according to a quantity of types of the cloud games run on the system containers in the target edge server, a quantity of shared directories to be provided, the network file system may also determine, according to the quantity of types of the cloud games run on the system containers in the target edge server, the quantity of shared directories to be provided.

[0040] When a quantity of types of cloud games run

on each system container in each edge server in the cloud gaming system is relatively small (that is, the quantity of types of the cloud games is less than the quantity threshold), the network file system may provide only one shared directory, so that the each system container in the cloud gaming system may be mounted to the shared directory. In this case, the target edge server may use the shared directory provided by the network file system as the target shared directory between the target container and the associated container. That is, in this implementation, FIG. 3d is a schematic diagram of mounting a target shared directory by a target container. As shown in FIG. 3d, since the network file system only includes the target shared directory, the target shared directory may be used for mounting the target container and the associated container of the target container and also used for mounting system containers in the cloud gaming system that are not related to the target container.

[0041] When the quantity of types of the cloud gaming run on each system container in each edge server in the cloud gaming system is relatively large (that is, the quantity of types of the cloud games is greater than or equal to the quantity threshold), the network file system may provide a plurality of shared directories. In this case, for related system containers running a same cloud game or related system containers of a cloud game sharing a same game engine, the related system containers are more likely to use a same GPU shader code, and therefore the related system containers may be mounted to a same shared directory. In addition, for unrelated system containers, the unrelated system containers may be mounted to other shared directories on the network file system. In this case, the target edge server may randomly select a shared directory from the plurality of shared directories provided by the network file system as the target shared directory between the target container and the associated container. That is, in this implementation, FIG. 3e is a schematic diagram of mounting a target shared directory by a target container. As shown in FIG. 3e, the network file system may include the target shared directory and also include other shared directories other than the target shared directory. Therefore, the target shared directory may only be used for mounting the target container and the associated container of the target container, and the other shared directories in the network file system may be used for mounting the system containers in the cloud gaming system that are not related to the target container. It is to be understood that, when there are a large quantity of cloud games, the network file system may provide only one shared directory if an indicator of the time consumed to search the target shared directory for a compilation result of a related GPU shader code is ignored.

[0042] In an actual application, the target edge server may randomly select either of the foregoing two selection manners to determine the target shared directory. Further, in the first selection manner, the target shared directory is selected from the server host, and both the target shared directory and the target container exist on the target edge server. Therefore, an access speed of the target container to each first compilation result in the target shared directory can be ensured. However, in the second manner, the target shared directory is selected from the network file system, and the network file system and the target edge server are independent of each other. Therefore, the access speed of the target container to each first compilation result in the target shared directory may be relatively low due to factors such as network congestion. In view of this, the target edge server may select one selection manner from the foregoing two selection manners according to an actual condition of the access speed of the network file system, to determine the target shared directory. Specifically, when the access speed of the network file system cannot be ensured, the first selection manner may be selected to determine the target shared directory; and when that the access speed of the network file system can be ensured, the second selection manner may be selected to determine the target shared directory.

[0043] It is to be understood that the selecting the selection manner according to a parameter such as the access speed of the network file system described in this embodiment of this application is merely exemplary rather than exhaustive. That is, in other embodiments, the selection manner may be selected according to other parameters. For example, since the first selection manner is to select the target shared directory from the server host, each first compilation result in the target shared directory cannot be accessed by system containers in other edge servers. However, since the second selection manner is to select the target shared directory from the network file system, the each first compilation result in the target shared directory can be accessed by the system containers in the other edge servers. In view of this, the target edge server may select one selection manner from the foregoing two selection manners according to an actual gaming deployment requirement, to determine the target shared directory. Specifically, when the gaming deployment requirement indicates that there is no need to share compilation results between different edge servers, the first selection manner may be selected to determine the target shared directory; and when the gaming deployment requirement indicates that there is a need to share compilation results between different edge servers, the second selection manner is selected to determine the target shared directory.

[0044] Further, based on the related description of the embodiments of the image processing method shown in FIG. 2, the specific process of producing the game mirror of the target game based on the static shared GPU shader cache mechanism, and the specific process of mounting the target shared directory based on the dynamic shared GPU shader cache mechanism, an embodiment of this application further provides a more specific schematic flowchart of an image processing method. In this embodiment of this application, the description is provid-

ed mainly by using an example in which the static shared GPU shader cache mechanism and the dynamic shared GPU shader cache mechanism are used in combination. The image processing method may be performed by the target container running in the target edge server in the cloud gaming system mentioned above. The target container is enabled by loading the game mirror of the target game, and the game mirror of the target container may include a second compilation result of at least one shader code compiled by pre-running the target game. Referring to FIG. 4, the image processing method may include the following steps S401 to S407.

[0045] S401: Receive a directory mounting instruction sent by the target edge server, the directory mounting instruction carrying the target shared directory.

[0046] S402: Designate a dynamic cache directory in the target container, and mount the target shared directory to the dynamic cache directory.

[0047] In an implementation, one directory may be designated in the target container as the dynamic cache directory. That is, in this implementation, the dynamic cache directory may be the same as or different from a storage directory of the second compilation result in the game mirror. In another implementation, when the target shared directory is mounted inside the target container, files in a corresponding directory inside the target shared directory to which the target shared directory is mounted may be cleared due to a limitation by a container directory mounting implementation. Therefore, to prevent the second compilation result in the game mirror from being cleared, two different environment variables may be used inside the target container to respectively designate the dynamic cache directory used for mounting the target shared directory and the storage directory of the second compilation result in the game mirror, so that the dynamic cache directory and the storage directory of the second compilation result are separated. In this way, when the target shared directory is mounted to the dynamic cache directory, only files in the dynamic cache directory are cleared, and the second compilation result in the storage directory is not affected.

[0048] In this implementation, the dynamic cache directory may be designated by using a first environment variable. The first environment variable may include, but is not limited to an environment variable "MESA_GLSL_CACHE_DIR" in a Mesa code. The storage directory of the second compilation result in the game mirror may be designated by using a second environment variable. Specifically, the second environment variable may include, but is not limited to an environment variable "MESA_GLSL_CACHE_DIR STATIC" in the Mesa code. Mesa is an open-source computer graphics library. Mesa may be used for implementing an application interface of opengl/opengles. Opengl is mainly a cross-platform application programming interface (API) used for rendering 3D graphics. Opengles is a subset of the opengl API and is mainly used in an embedded device (such as a smartphone).

[0049] After the dynamic cache directory is designated in the target container, the target shared directory may be mounted to the dynamic cache directory, to keep data between the target shared directory and the dynamic cache directory synchronized. The target shared directory is also mounted to a preset directory of the associated container, to cache the first compilation result obtained by compiling, when running the associated game, at least one shader code by the associated container. For example, when the target shared directory is located on the server host, if the server host only includes the target shared directory, reference may be made to FIG. 5a for a schematic diagram of mounting the target shared directory; and if the server host includes the target shared directory and other shared directories, reference may be made to FIG. 5b for a schematic diagram of mounting the target shared directory. In another example, when the target shared directory is located on the network file system, if the network file system only includes the target shared directory, reference may be made to FIG. 5c for a schematic diagram of mounting the target shared directory; and if the network file system includes the target shared directory and other shared directories, reference may be made to FIG. 5d for a schematic diagram of mounting the target shared directory.

[0050] S403: Obtain, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event.

[0051] S404: Detect whether a second compilation result of the target shader code exists in the game mirror of the target game.

[0052] Based on the above, the game mirror may include a second compilation result of at least one shader code compiled by pre-running the target game. In an implementation, the game mirror may further include a shader code corresponding to each second compilation result. In this implementation, it may be directly detected whether the target shader code is included in the game mirror of the target game. It may be determined that the second compilation result of the target shader code exists in the game mirror when the target shader code is included in the game mirror; and it may be determined that the second compilation result of the target shader code does not exist in the game mirror when the target shader code is not included in the game mirror.

[0053] In another implementation, the game mirror may further include an index value of at least one shader code, and each index value corresponds to one second compilation result. Therefore, in this implementation, a hash operation may be performed on the target shader code, to obtain a target index value of the target shader code. Then, it is detected whether the target index value is included in the game mirror. It may be determined that the second compilation result of the target shader code exists in the game mirror when the target index value is included in the game mirror; and it may be determined that the second compilation result of the target shader code does not exist in the game mirror when the target

index value is not included in the game mirror.

[0054] Step S405 may be performed when the second compilation result of the target shader code exists in the game mirror; and step S406 may be performed when the second compilation result of the target shader code does not exist in the game mirror.

[0055] Step S405: Obtain the second compilation result of the target shader code from the game mirror as the target compilation result when the second compilation result exists.

[0056] Step S406: Obtain the first compilation result from the target shared directory between the target container and the associated container of the target container as the target compilation result of the target shader code when the second compilation result does not exist.

[0057] In a specific implementation, the target shared directory may include a first compilation result of at least one shader code, and each first compilation result may correspond to one index value. Reference may be made to FIG. 5e together for a specific implementation of step S406. Specifically, the target container may search the target shared directory to determine whether a target index value exists. When the target index value exists, a code compilation process may be skipped to directly obtain a first compilation result corresponding to the target index value as the target compilation result. When the target shader code does not exist, it indicates that the game resource does not include the target compilation result of the target shader code. Therefore, the target shader code may be compiled to obtain the target compilation result; and the target compilation result is cached to the dynamic cache directory, to synchronize the target compilation result to the target shared directory, so that the associated container directly obtains the target compilation result from the target shared directory when the target shader code needs to be compiled. In addition, the target index value of the target shader code may be cached to the dynamic cache directory together, to synchronize the target index value to the target shared directory.

[0058] S407: Perform graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event. After step S407 is performed, the image processing method may further include: displaying the feedback image on the target game client.

[0059] In this embodiment of this application, after a target shader code corresponding to the operation event is obtained in response to an operation event of a target game running in a target container, a compilation process of the target shader code may be skipped to directly obtain a target compilation result of the target shader code from a game mirror or a target shared directory between the target container and an associated container of the target container. Then, graphic processing may be performed according to the target compilation result, to obtain a feedback image corresponding to the operation event; and the feedback image is displayed on a target game client. It can be seen that, in this embodiment of this application, graphic processing is performed by directly using the target compilation result compiled from the game mirror or the associated container in the target shared directory, so that a duration of compiling the target shader code can be effectively shortened, thereby increasing the running speed of a target game.

[0060] Based on the foregoing description of the embodiments of the image processing method, an embodiment of this application further discloses an image processing apparatus. The image processing apparatus may be a computer program (including program code) run in the target edge server. The image processing apparatus may perform the method shown in FIG. 2 or FIG. 4. Referring to FIG. 6, the image processing apparatus may operate the following units:

an obtaining unit 601, configured to obtain, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event, the operation event being detected in a game picture of the target game displayed on a target game client that is connected to the target container; and

the obtaining unit 601 being further configured to obtain a target compilation result of the target shader code from a game resource, the game resource including at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared directory including a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and

a processing unit 602, configured to: perform graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event; and display the feedback image on the target game client.

[0061] In an implementation, the game resource includes the game mirror and the target shared directory; the game mirror includes a second compilation result of at least one shader code compiled by pre-running the target game, and each second compilation result is added to the game mirror in a process of producing the game mirror; and Correspondingly, when the obtaining unit 601 is configured to obtain the target compilation result of the target shader code from the game resource, the obtaining unit 601 may further be configured to:

detect whether a second compilation result of the target shader code exists in the game mirror of the target game;

obtain the second compilation result of the target

shader code from the game mirror as the target compilation result when the second compilation result exists; and

obtain the first compilation result from the target shared directory between the target container and the associated container of the target container as the target compilation result of the target shader code when the second compilation result does not exist.

[0062] In another implementation, the game mirror further includes an index value of at least one shader code, and each index value corresponds to one second compilation result. Correspondingly, when the obtaining unit 601 is configured to detect whether a second compilation result of the target shader code exists in the game mirror of the target game, the obtaining unit 601 may further be configured to:

perform a hash operation on the target shader code, to obtain a target index value of the target shader code;

determine that the second compilation result of the target shader code exists in the game mirror when the target index value is included in the game mirror; and

determine that the second compilation result of the target shader code does not exist in the game mirror when the target index value is not included in the game mirror.

[0063] In another implementation, the target game is a cloud game in a cloud gaming system, and the cloud gaming system includes at least one edge server and a plurality of game clients; at least one system container is deployed inside each edge server, and each system container is connected to at least one game client; and

the each system container is configured to run one or more games, and in a process of running any game, the each system container transmits a game picture of the any game to the game client to which the system container is connected for display, where

the target container is any system container running based on a server operating system of a target edge server, and the target edge server is any edge server in the cloud gaming system; and the associated container is a system container running an associated game of the target game, and the associated game includes: the target game or a game using a same game engine as the target game.

[0064] In another implementation, the processing unit 602 may further be configured to:

receive a directory mounting instruction transmitted by the target edge server, the directory mounting instruction carrying the target shared directory; and

designate a dynamic cache directory in the target container, and mount the target shared directory to the dynamic cache directory, to keep data between the target shared directory and the dynamic cache directory synchronized, where

the target shared directory is also mounted to a preset directory of the associated container, to cache the first compilation result obtained by compiling, when running the associated game, at least one shader code by the associated container.

[0065] In another implementation, the dynamic cache directory is designated by using a first environment variable, and the second compilation result is designated in a storage directory of the game mirror by using a second environment variable.

[0066] In another implementation, the target shared directory is located in a server host on which the server operating system resides. Alternatively,
the target shared directory is located in a network file system, and the network file system is independent of each edge server in the cloud gaming system and allows assess by a system container of the each edge server.

[0067] In another implementation, when the target shared directory is located in the server host, the server host further includes other shared directories other than the target shared directory, and the other shared directories in the server host are used for mounting system containers in the target edge server that are not related to the target container.

[0068] When the target shared directory is located in the network file system, the network file system further includes other shared directories other than the target shared directory, and the other shared directories in the network file system are used for mounting system containers in the cloud gaming system that are not related to the target container.

[0069] In another implementation, the game resource does not include the target compilation result of the target shader code. Correspondingly, the processing unit 602 may further be configured to:

when the target compilation result fails to be obtained from the game resource, compile the target shader code to obtain the target compilation result; and

cache the target compilation result to the dynamic cache directory, to synchronize the target compilation result to the target shared directory, so that the associated container obtains the target compilation result from the target shared directory when the target shader code needs to be compiled.

[0070] According to an embodiment of this application, the steps involved in the method shown in FIG. 2 or FIG. 4 may be performed by the units of the image processing apparatus shown in FIG. 6. For example, steps S201 and S202 shown in FIG. 2 may be performed by the obtaining unit 601 shown in FIG. 6, and step S203 may be performed by the processing unit 602 shown in FIG. 6. In another example, steps S401, S402, and S407 shown in FIG. 4 may be performed by the processing unit 602 shown in FIG. 6, and steps S403 to S406 may be performed by the obtaining unit 601 shown in FIG. 6.

[0071] According to another embodiment of this application, the units of the image processing apparatus shown in FIG. 6 may be separately or wholly combined into one or several other units, or one (or more) of the units may further be divided into a plurality of units of smaller functions. In this way, same operations may be implemented, and the implementation of the technical effects of the embodiments of this application is not affected. The foregoing units are divided based on logical functions. In an actual application, a function of one unit may also be implemented by a plurality of units, or functions of a plurality of units are implemented by one unit. In other embodiments of this application, the image processing apparatus may also include other units. In an actual application, the functions may also be cooperatively implemented by other units and may be cooperatively implemented by a plurality of units.

[0072] According to another embodiment of this application, a computer program (including program code) that can perform the steps in the corresponding method shown in FIG. 2 or FIG. 4 may be run on a general-purpose computing device such as a computer, which includes processing elements and storage elements such as a CPU, a random access memory (RAM), and a read-only memory (ROM), to construct the image processing apparatus shown in FIG. 6, and implement the image processing method in the embodiments of this application. The computer program may be recorded in, for example, a computer-readable storage medium, and may be loaded into the foregoing computing device by using the computer-readable storage medium, and run in the computing device.

[0073] In this embodiment of this application, after a target shader code corresponding to the operation event is obtained in response to an operation event of a target game running in a target container, a compilation process of the target shader code may be skipped to directly obtain a target compilation result of the target shader code from a game mirror or a target shared directory between the target container and an associated container of the target container. Then, graphic processing may be performed according to the target compilation result, to obtain a feedback image corresponding to the operation event; and the feedback image is displayed on a target game client. It can be seen that, in this embodiment of this application, graphic processing is performed by directly using the target compilation result compiled from the game mirror or the associated container in the target shared directory, so that a duration of compiling the target shader code can be effectively shortened, thereby increasing the running speed of a target game.

[0074] Based on the descriptions of the foregoing method embodiments and apparatus embodiments, an embodiment of this application further provides a server. The server may be the target edge server mentioned above. Referring to FIG. 7, the server includes at least a processor 701, an input interface 702, an output interface 703, and a computer storage medium 704. The processor 701, the input interface 702, the output interface 703, and the computer storage medium 704 in the server may be connected by using a bus or in another manner.

[0075] The computer storage medium 704 is a memory device in the server and is configured to store programs and data. It may be understood that, the computer storage medium 704 may include an internal storage medium of the server and may also include an expanded storage medium supported by the server. The computer storage medium 704 provides a storage space storing an operating system of the server. In addition, the storage space further stores one or more instructions adapted to be loaded and executed by the processor 701. The instructions may be one or more computer programs (including program code). The computer storage medium may be a high-speed RAM, or may be a non-volatile memory, for example, at least one magnetic disk memory. Alternatively, the computer storage medium may also be at least one computer storage medium far away from the foregoing processor. The processor 701 (or referred to as a CPU) is a computing core and a control core of the server, which is adapted to implement one or more instructions, and specifically, adapted to load and execute one or more instructions to implement corresponding method processes or corresponding functions.

[0076] In another embodiment, the processor 701 may load and execute one or more instructions stored in the computer storage medium 704, to implement corresponding steps of the embodiments of the image processing method shown in FIG. 2 or FIG. 4. In a specific implementation, the one or more instructions in the computer storage medium 704 are loaded by the processor 701 to further perform the following steps:

obtaining, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event, where the operation event is detected in a game picture of the target game displayed on a target game client that is connected to the target container;

obtaining a target compilation result of the target shader code from a game resource, the game resource including at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared

directory including a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and

performing graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event; and displaying the feedback image on the target game client.

[0077] In an implementation, the game resource includes the game mirror and the target shared directory; the game mirror includes a second compilation result of at least one shader code compiled by pre-running the target game, and each second compilation result is added to the game mirror in a process of producing the game mirror; and Correspondingly, in a process of obtaining a target compilation result of the target shader code from a game resource, the one or more instructions may be loaded by the processor 701 to specifically perform the following steps:

detecting whether a second compilation result of the target shader code exists in the game mirror of the target game;

obtaining the second compilation result of the target shader code from the game mirror as the target compilation result when the second compilation result exists; and

obtaining the first compilation result from the target shared directory between the target container and the associated container of the target container as the target compilation result of the target shader code when the second compilation result does not exist.

[0078] In another implementation, the game mirror further includes an index value of at least one shader code, and each index value corresponds to one second compilation result. Correspondingly, in a process of detecting whether a second compilation result of the target shader code exists in the game mirror of the target game, the one or more instructions may be loaded by the processor 701 to specifically perform the following steps:

performing a hash operation on the target shader code, to obtain a target index value of the target shader code;

determining that the second compilation result of the target shader code exists in the game mirror when the target index value is included in the game mirror; and

determining that the second compilation result of the target shader code does not exist in the game mirror when the target index value is not included in the game mirror.

[0079] In another implementation, the target game is a cloud game in a cloud gaming system, and the cloud gaming system includes at least one edge server and a plurality of game clients; at least one system container is deployed inside each edge server, and each system container is connected to at least one game client; and

the each system container is configured to run one or more games, and in a process of running any game, the each system container transmits a game picture of the any game to the game client to which the system container is connected for display, where

the target container is any system container running based on a server operating system of a target edge server, and the target edge server is any edge server in the cloud gaming system; and the associated container is a system container running an associated game of the target game, and the associated game includes: the target game or a game using a same game engine as the target game.

[0080] In another implementation, the one or more instructions may further be loaded by the processor 701 to perform the following steps:

receiving a directory mounting instruction transmitted by the target edge server, the directory mounting instruction carrying the target shared directory; and

designating a dynamic cache directory in the target container, and mounting the target shared directory to the dynamic cache directory, to keep data between the target shared directory and the dynamic cache directory synchronized, where

the target shared directory is also mounted to a preset directory of the associated container, to cache the first compilation result obtained by compiling, when running the associated game, at least one shader code by the associated container.

[0081] In another implementation, the dynamic cache directory is designated by using a first environment variable, and the second compilation result is designated in a storage directory of the game mirror by using a second environment variable.

[0082] In another implementation, the target shared directory is located in a server host on which the server operating system resides. Alternatively,
the target shared directory is located in a network file system, and the network file system is independent of each edge server in the cloud gaming system and allows assess by a system container of the each edge server.

[0083] In another implementation, when the target shared directory is located in the server host, the server

host further includes other shared directories other than the target shared directory, and the other shared directories in the server host are used for mounting system containers in the target edge server that are not related to the target container.

[0084] When the target shared directory is located in the network file system, the network file system further includes other shared directories other than the target shared directory, and the other shared directories in the network file system are used for mounting system containers in the cloud gaming system that are not related to the target container.

[0085] In another implementation, the game resource does not include the target compilation result of the target shader code. Correspondingly, the one or more instructions may further be loaded by the processor 701 to perform the following steps:

when the target compilation result fails to be obtained from the game resource, compiling the target shader code to obtain the target compilation result; and

caching the target compilation result to the dynamic cache directory, to synchronize the target compilation result to the target shared directory, so that the associated container obtains the target compilation result from the target shared directory when the target shader code needs to be compiled.

[0086] In this embodiment of this application, after a target shader code corresponding to the operation event is obtained in response to an operation event of a target game running in a target container, a compilation process of the target shader code may be skipped to directly obtain a target compilation result of the target shader code from a game mirror or a target shared directory between the target container and an associated container of the target container. Then, graphic processing may be performed according to the target compilation result, to obtain a feedback image corresponding to the operation event; and the feedback image is displayed on a target game client. It can be seen that, in this embodiment of this application, graphic processing is performed by directly using the target compilation result compiled from the game mirror or the associated container in the target shared directory, so that a duration of compiling the target shader code can be effectively shortened, thereby increasing the running speed of a target game.

[0087] According to an aspect of the embodiments of this application, a computer program product or a computer program is further provided, the computer program product or the computer program including computer instructions, the computer instructions being stored in a computer-readable storage medium. The processor of the computer device reads the computer instructions from the computer-readable storage medium, and the processor executes the computer instructions, to cause the computer device to perform the steps of the foregoing embodiments of the image processing method shown in FIG. 2 or FIG. 4.

[0088] It is to be understood that, the description disclosed above is merely exemplary embodiments of this application, and certainly is not intended to limit the scope of the claims of this application. Therefore, equivalent variations made according to the claims of this application shall fall within the scope of this application.

Claims

1. An image processing method, comprising:

obtaining, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event;

obtaining a target compilation result of the target shader code from a game resource, the game resource comprising at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared directory comprising a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and

performing graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event.

2. The method according to claim 1, wherein the game resource comprises the game mirror and the target shared directory; the game mirror comprises a second compilation result of at least one shader code compiled by pre-running the target game, and each second compilation result is added to the game mirror in a process of producing the game mirror; and the obtaining a target compilation result of the target shader code from a game resource comprises:

detecting whether a second compilation result of the target shader code exists in the game mirror of the target game;

obtaining the second compilation result of the target shader code from the game mirror as the target compilation result when the second compilation result exists; and

obtaining the first compilation result from the target shared directory between the target container and the associated container of the target container as the target compilation result of the target shader code when the second compilation result does not exist.

3. The method according to claim 2, wherein the game mirror further comprises an index value of at least

one shader code, and each index value corresponds to one second compilation result; and the detecting whether a second compilation result of the target shader code exists in the game mirror of the target game comprises:

performing a hash operation on the target shader code, to obtain a target index value of the target shader code;
determining that the second compilation result of the target shader code exists in the game mirror when the game mirror comprises the target index value; and
determining that the second compilation result of the target shader code does not exist in the game mirror when the game mirror does not comprise the target index value.

4. The method according to claim 1, wherein the target game is a cloud game in a cloud gaming system, and the cloud gaming system comprises at least one edge server and a plurality of game clients; at least one system container is deployed inside each edge server, and each system container is connected to at least one game client; and

the each system container is configured to run one or more games, and in a process of running any game, the each system container transmits a game picture of the any game to the game client to which the system container is connected for display, wherein
the target container is a system container running based on a server operating system of a target edge server, and the target edge server is any edge server in the cloud gaming system; and the associated container is a system container running an associated game of the target game, and the associated game comprises: the target game or a game using a same game engine as the target game.

5. The method according to claim 4, further comprising:

receiving a directory mounting instruction transmitted by the target edge server, the directory mounting instruction carrying the target shared directory; and
designating a dynamic cache directory in the target container, and mounting the target shared directory to the dynamic cache directory, to keep data between the target shared directory and the dynamic cache directory synchronized, wherein
the target shared directory is mounted to a preset directory of the associated container, to cache the first compilation result obtained by compiling, when running the associated game, at least one shader code by the associated container.

6. The method according to claim 5, wherein the dynamic cache directory is designated by using a first environment variable, and the second compilation result is designated in a storage directory of the game mirror by using a second environment variable.

7. The method according to any one of claims 4 to 6, wherein the target shared directory is located in a server host on which the server operating system resides.

8. The method according to any one of claims 4 to 6, wherein the shared directory is located in a network file system, and the network file system is independent of each edge server in the cloud gaming system and is accessible to a system container of each edge server.

9. The method according to claim 7, wherein the server host further comprises other shared directories distinct from than the target shared directory, and the other shared directories in the server host are used for mounting system containers in the target edge server that are not related to the target container.

10. The method according to claim 8, wherein the network file system further comprises shared directories distinct from the target shared directory, and the other shared directories in the network file system are used for mounting system containers in the cloud gaming system that are not related to the target container.

11. The method according to claim 5, wherein the game resource does not comprise the target compilation result of the target shader code, and the method further comprises:

when the target compilation result fails to be obtained from the game resource, compiling the target shader code to obtain the target compilation result; and
caching the target compilation result to the dynamic cache directory, to synchronize the target compilation result to the target shared directory, wherein the associated container obtains the target compilation result from the target shared directory when the target shader code needs to be compiled.

12. The method according to claim 1, wherein the operation event is detected in a game picture of the target game displayed on a target game client that is connected to the target container, and after the feedback image corresponding to the operation event is ob-

tained, the method further comprises:
displaying the feedback image on the target game client.

13. An image processing apparatus, comprising:

an obtaining unit, configured to obtain, in response to an operation event of a target game running in a target container, a target shader code corresponding to the operation event,
the obtaining unit being further configured to obtain a target compilation result of the target shader code from a game resource, the game resource comprising at least one of the following: a game mirror of the target game and a target shared directory between the target container and an associated container of the target container, the target shared directory comprising a first compilation result obtained by compiling, when running the game, at least one shader code by the associated container; and
a processing unit, configured to perform graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event.

14. The apparatus according to claim 13, wherein the game resource comprises the game mirror and the target shared directory; the game mirror comprises a second compilation result of at least one shader code compiled by pre-running the target game, and each second compilation result is added to the game mirror in a process of producing the game mirror; and the obtaining unit is further configured to:

detect whether a second compilation result of the target shader code exists in the game mirror of the target game;
obtain the second compilation result of the target shader code from the game mirror as the target compilation result when the second compilation result exists; and
obtain the first compilation result from the target shared directory between the target container and the associated container of the target container as the target compilation result of the target shader code when the second compilation result does not exist.

15. The apparatus according to claim 14, wherein the game mirror further comprises an index value of at least one shader code, and each index value corresponds to one second compilation result; and
the obtaining unit is further configured to:

perform a hash operation on the target shader code, to obtain a target index value of the target shader code;
determine that the second compilation result of the target shader code exists in the game mirror when the game mirror comprises the target index value; and
determine that the second compilation result of the target shader code does not exist in the game mirror when the game mirror does not comprise the target index value.

16. The apparatus according to claim 13, wherein the target game is a cloud game in a cloud gaming system, and the cloud gaming system comprises at least one edge server and a plurality of game clients; at least one system container is deployed inside each edge server, and each system container is connected to at least one game client; and

the each system container is configured to run one or more games, and in a process of running any game, the each system container transmits a game picture of the any game to the game client to which the system container is connected for display, wherein
the target container is a system container running based on a server operating system of a target edge server, and the target edge server is any edge server in the cloud gaming system; and the associated container is a system container running an associated game of the target game, and the associated game comprises: the target game or a game using a same game engine as the target game.

17. The apparatus according to claim 16, wherein the processing unit is further configured to:

receive a directory mounting instruction transmitted by the target edge server, the directory mounting instruction carrying the target shared directory; and
designate a dynamic cache directory in the target container, and mount the target shared directory to the dynamic cache directory, to keep data between the target shared directory and the dynamic cache directory synchronized, wherein
the target shared directory is mounted to a preset directory of the associated container, to cache the first compilation result obtained by compiling, when running the game, at least one shader code by the associated container.

18. The apparatus according to claim 13, wherein the operation event is detected in a game picture of the target game displayed on a target game client that is connected to the target container, and the processing unit is further configured to:
display the feedback image on the target game cli-

ent.

19. A server, comprising an input interface and an output interface, the server further comprising:

a computer storage medium, configured to store one or more instructions; and
a processor, configured to load the one or more instructions stored in the computer storage medium, to perform the image processing method according to any one of claims 1 to 12.

20. A computer storage medium, storing one or more instructions, the one or more instructions being configured to be loaded by a processor to perform the image processing method according to any one of claims 1 to 12.

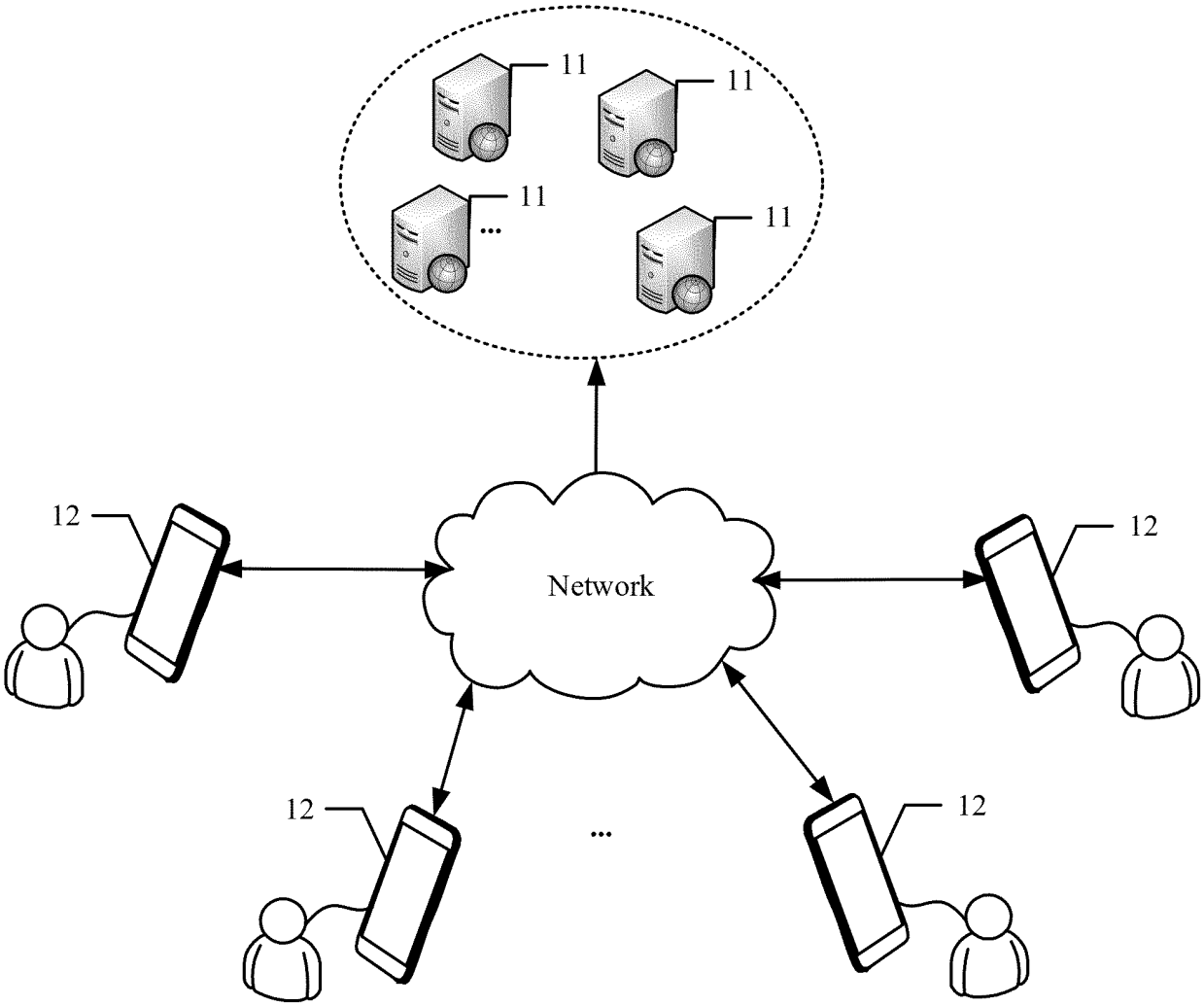

FIG. 1a

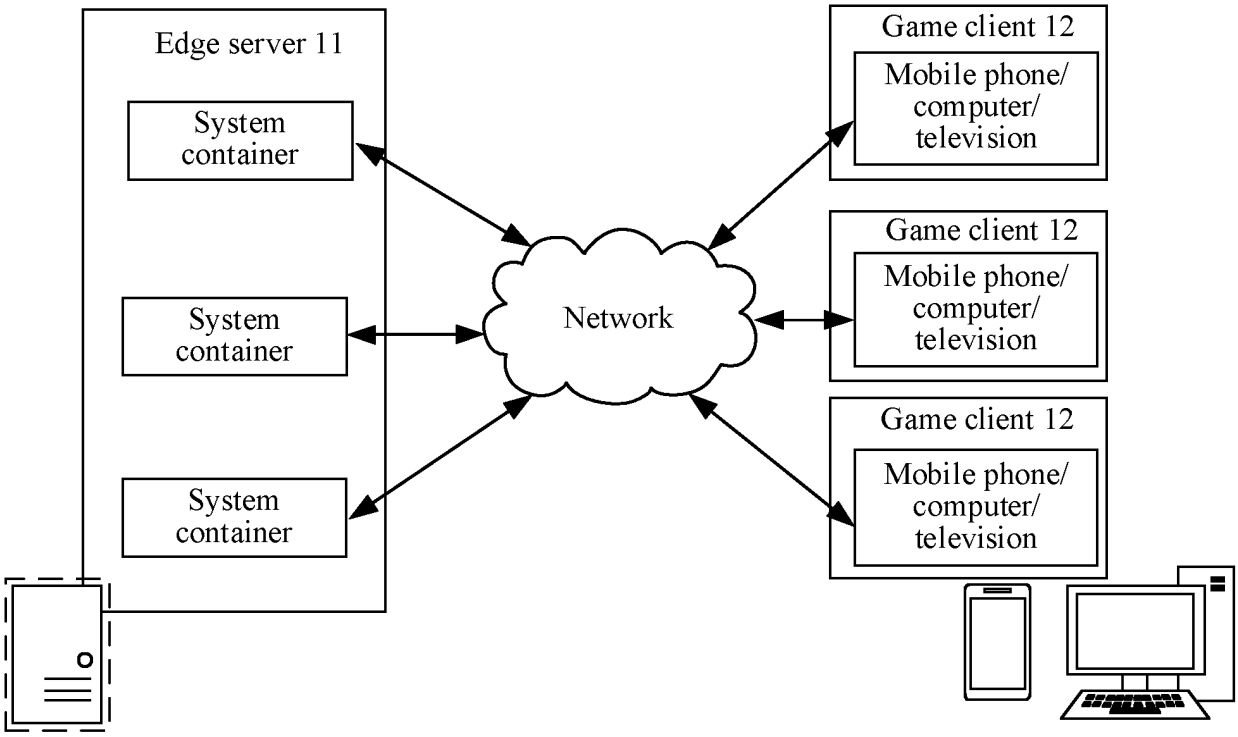
Edge server 11
System container
System container
System container
Network
Game client 12
Mobile phone/ computer/ television
Game client 12
Mobile phone/ computer/ television
Game client 12
Mobile phone/ computer/ television

FIG. 1b

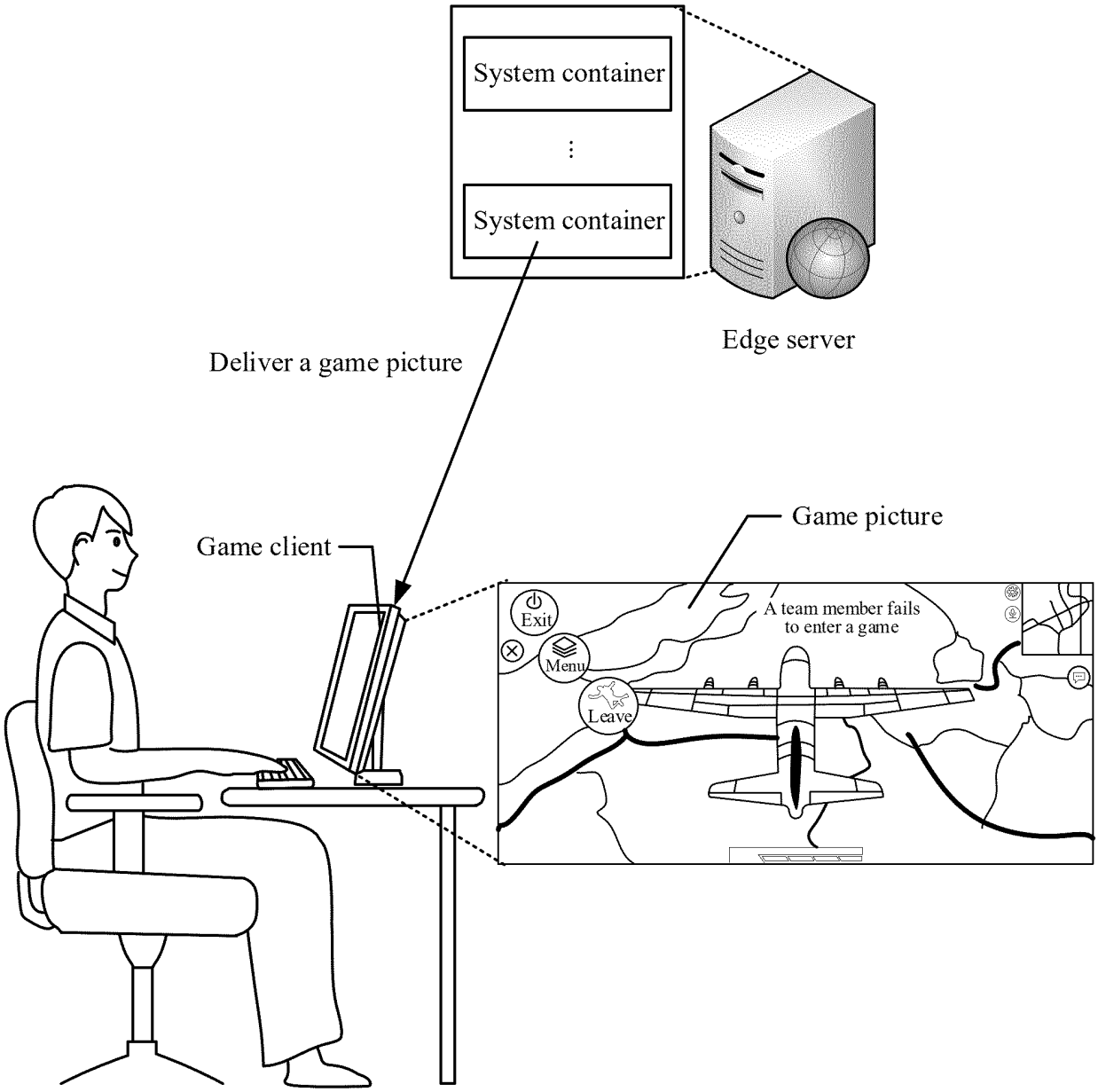
System container
System container
Edge server
Deliver a game picture
Game client
Game picture
Exit
Menu
Leave
A team member fails to enter a game

FIG. 1c

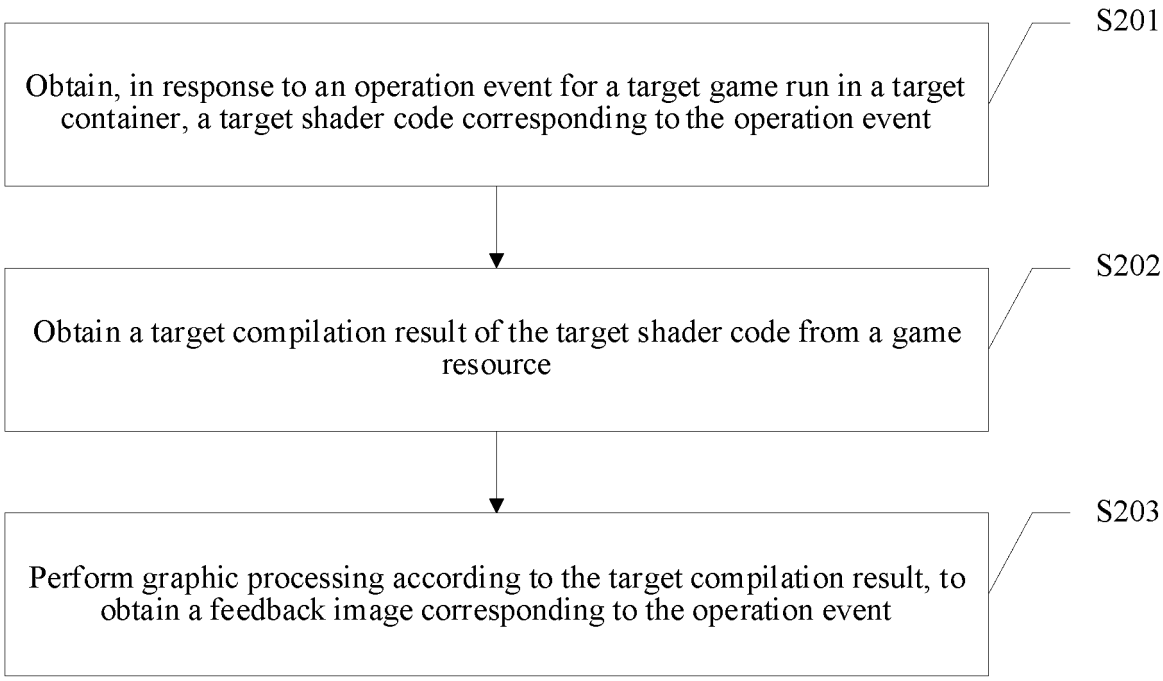
Obtain, in response to an operation event for a target game run in a target container, a target shader code corresponding to the operation event
S201
Obtain a target compilation result of the target shader code from a game resource
S202
Perform graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event
S203

FIG. 2

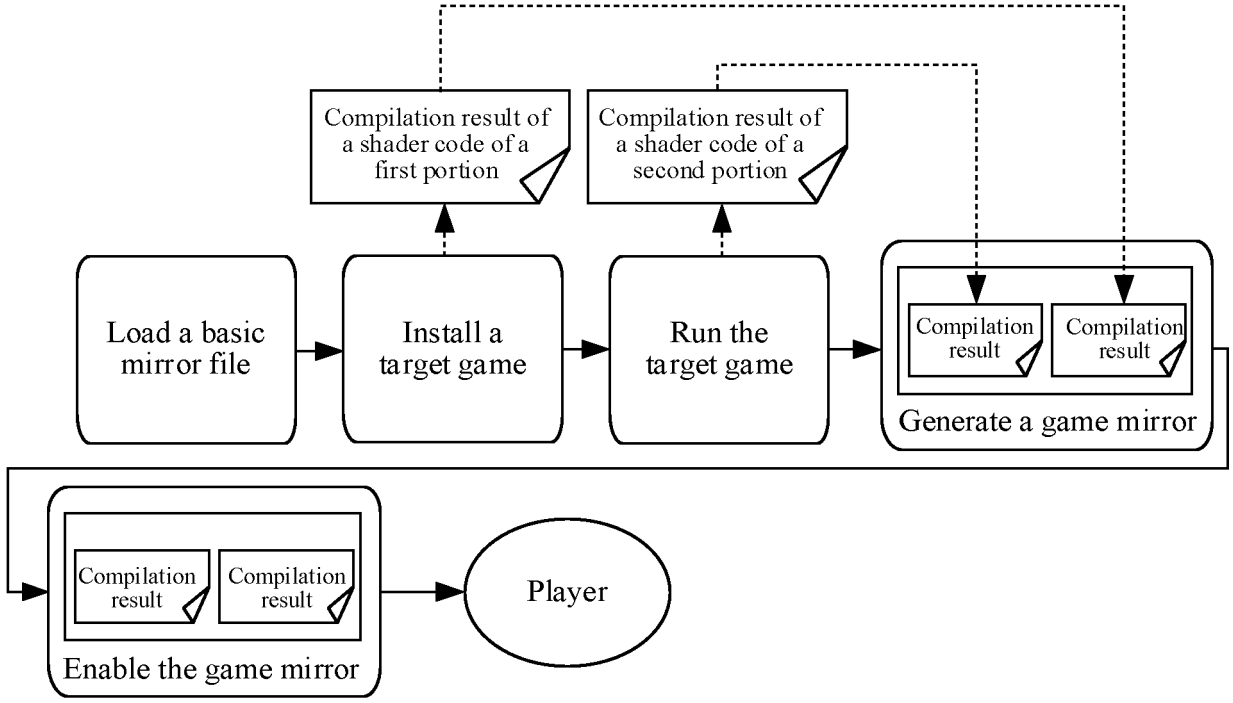
Compilation result of a shader code of a first portion
Compilation result of a shader code of a second portion
Load a basic mirror file
Install a target game
Run the target game
Compilation result
Compilation result
Generate a game mirror
Compilation result
Compilation result
Enable the game mirror
Player

FIG. 3a

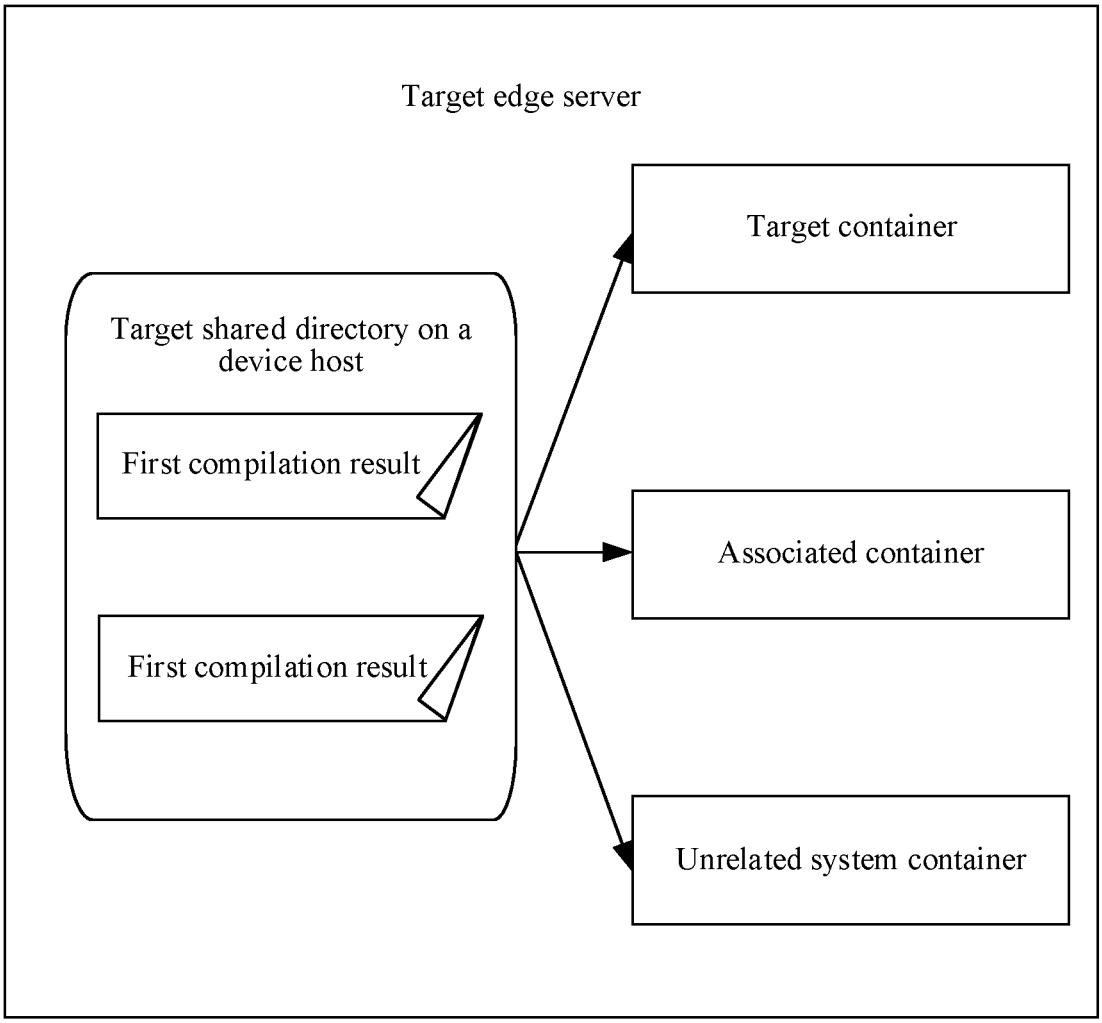
Target edge server
Target container
Target shared directory on a device host
First compilation result
First compilation result
Associated container
Unrelated system container

FIG. 3b

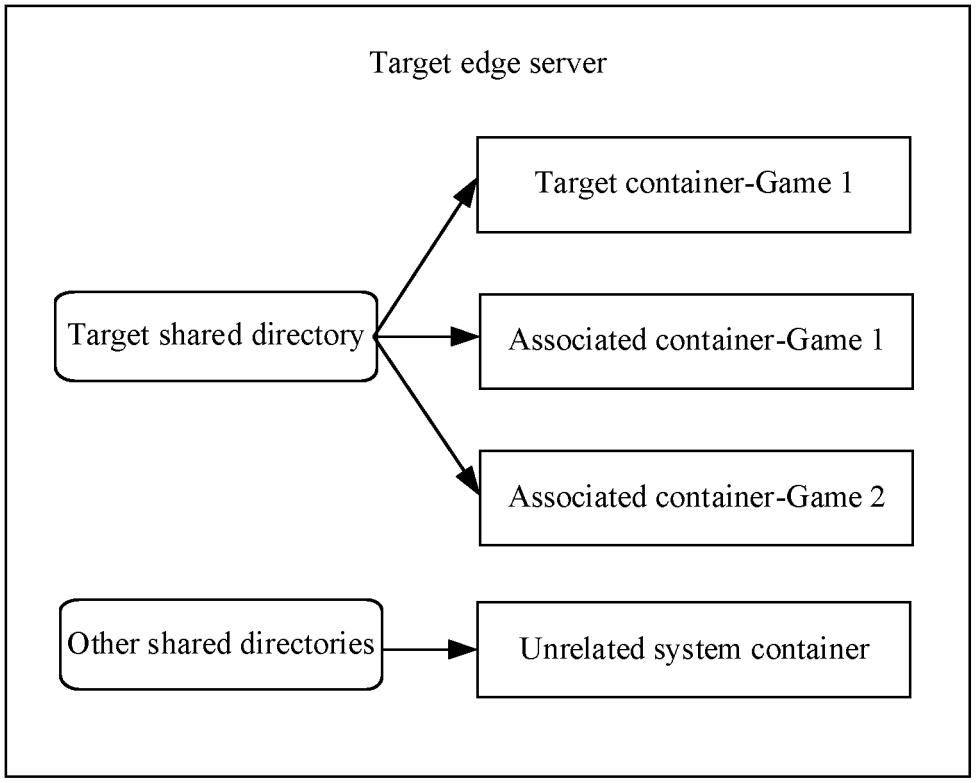
Target edge server
Target container-Game 1
Target shared directory
Associated container-Game 1
Associated container-Game 2
Other shared directories
Unrelated system container

FIG. 3c

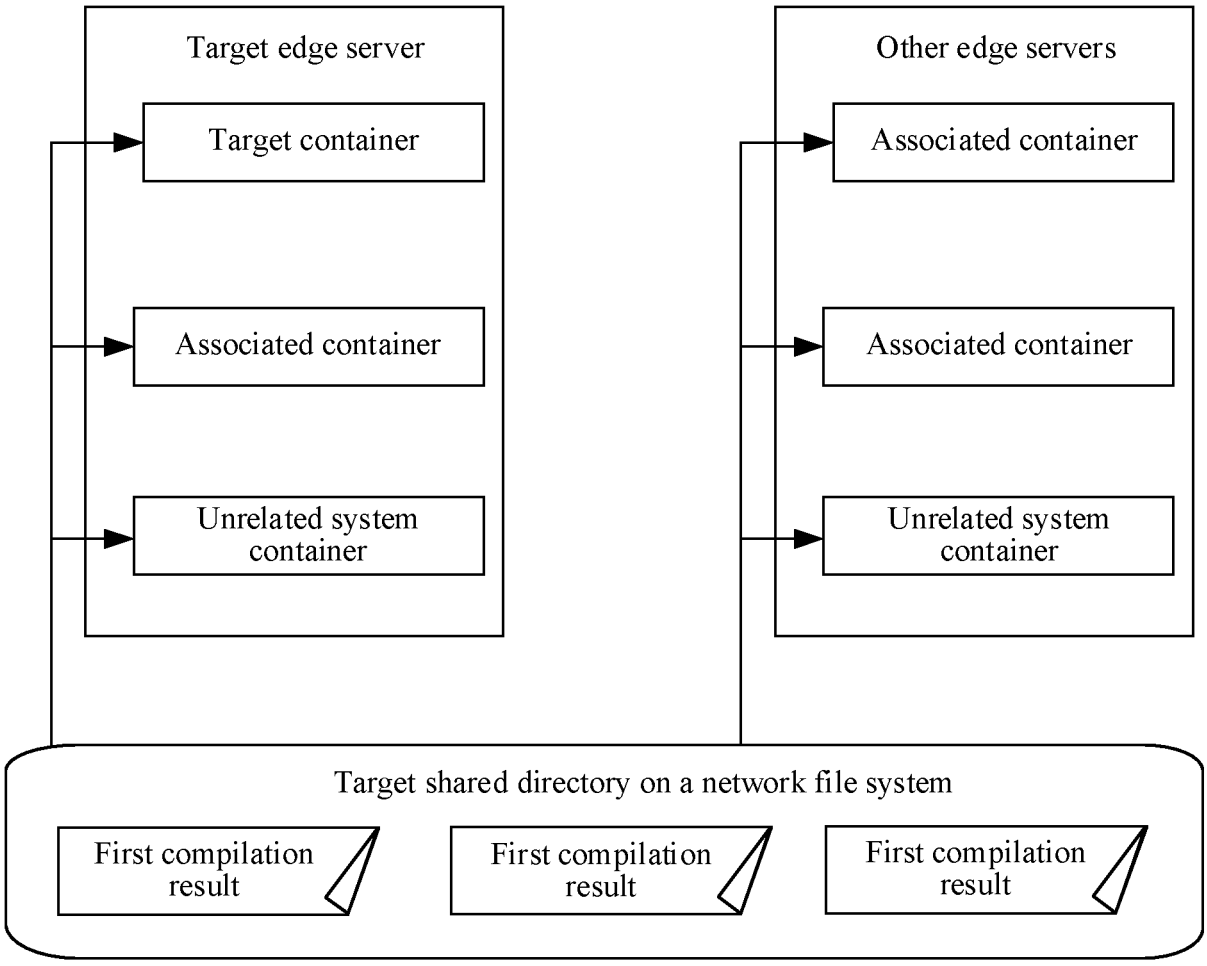
Target edge server
Target container
Associated container
Unrelated system container
Other edge servers
Associated container
Associated container
Unrelated system container
Target shared directory on a network file system
First compilation result
First compilation result
First compilation result

FIG. 3d

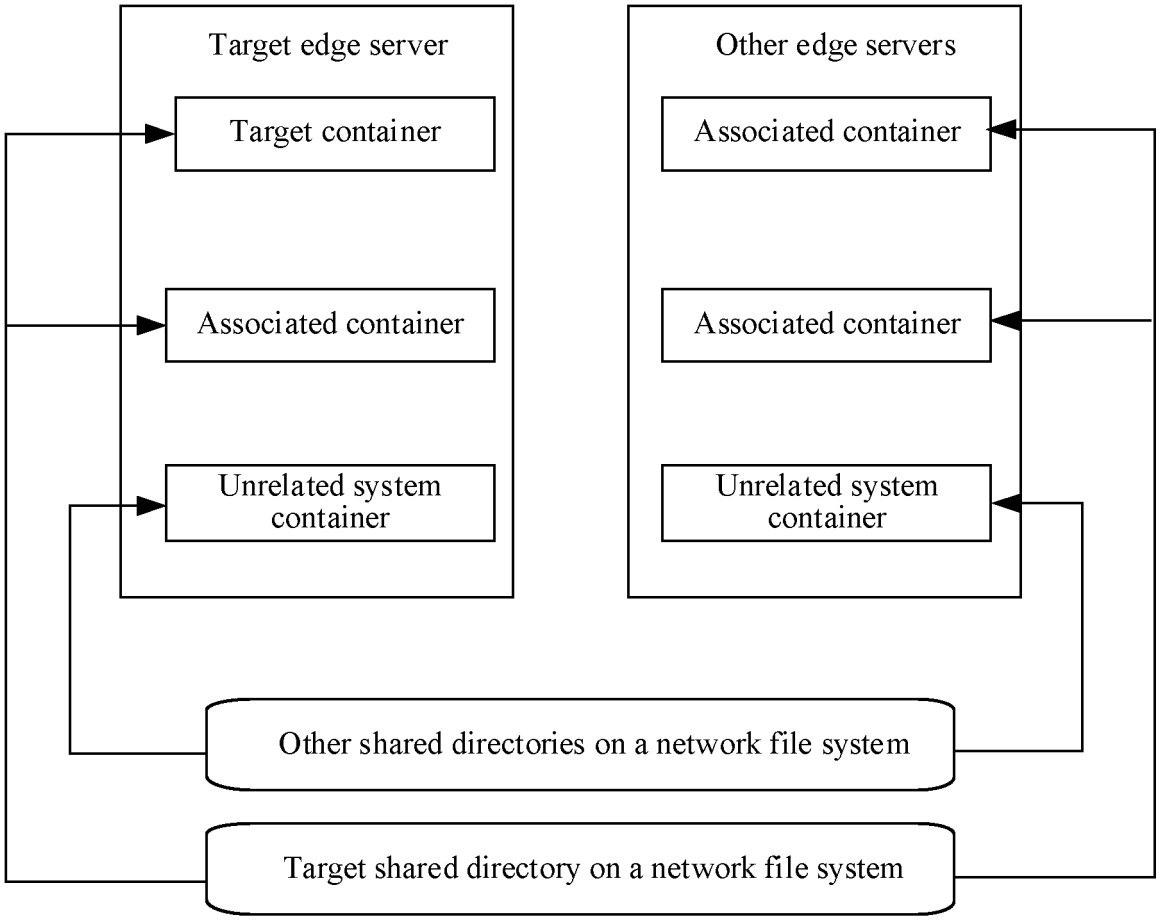
Target edge server
Target container
Associated container
Unrelated system container
Other edge servers
Associated container
Associated container
Unrelated system container
Other shared directories on a network file system
Target shared directory on a network file system

FIG. 3e

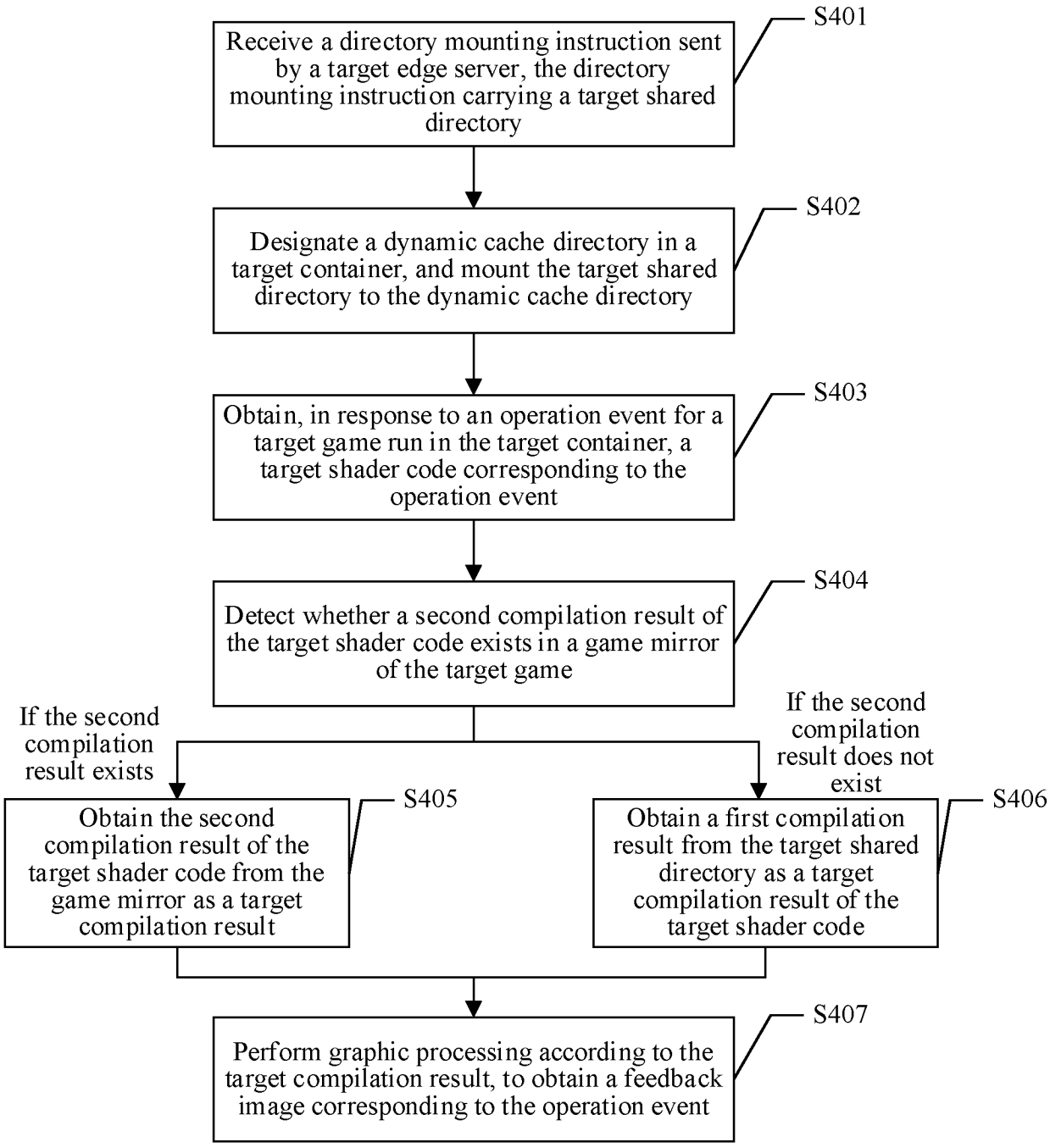
S401
Receive a directory mounting instruction sent by a target edge server, the directory mounting instruction carrying a target shared directory
S402
Designate a dynamic cache directory in a target container, and mount the target shared directory to the dynamic cache directory
S403
Obtain, in response to an operation event for a target game run in the target container, a target shader code corresponding to the operation event
S404
Detect whether a second compilation result of the target shader code exists in a game mirror of the target game
If the second compilation result exists
If the second compilation result does not exist
S405
Obtain the second compilation result of the target shader code from the game mirror as a target compilation result
S406
Obtain a first compilation result from the target shared directory as a target compilation result of the target shader code
S407
Perform graphic processing according to the target compilation result, to obtain a feedback image corresponding to the operation event

FIG. 4

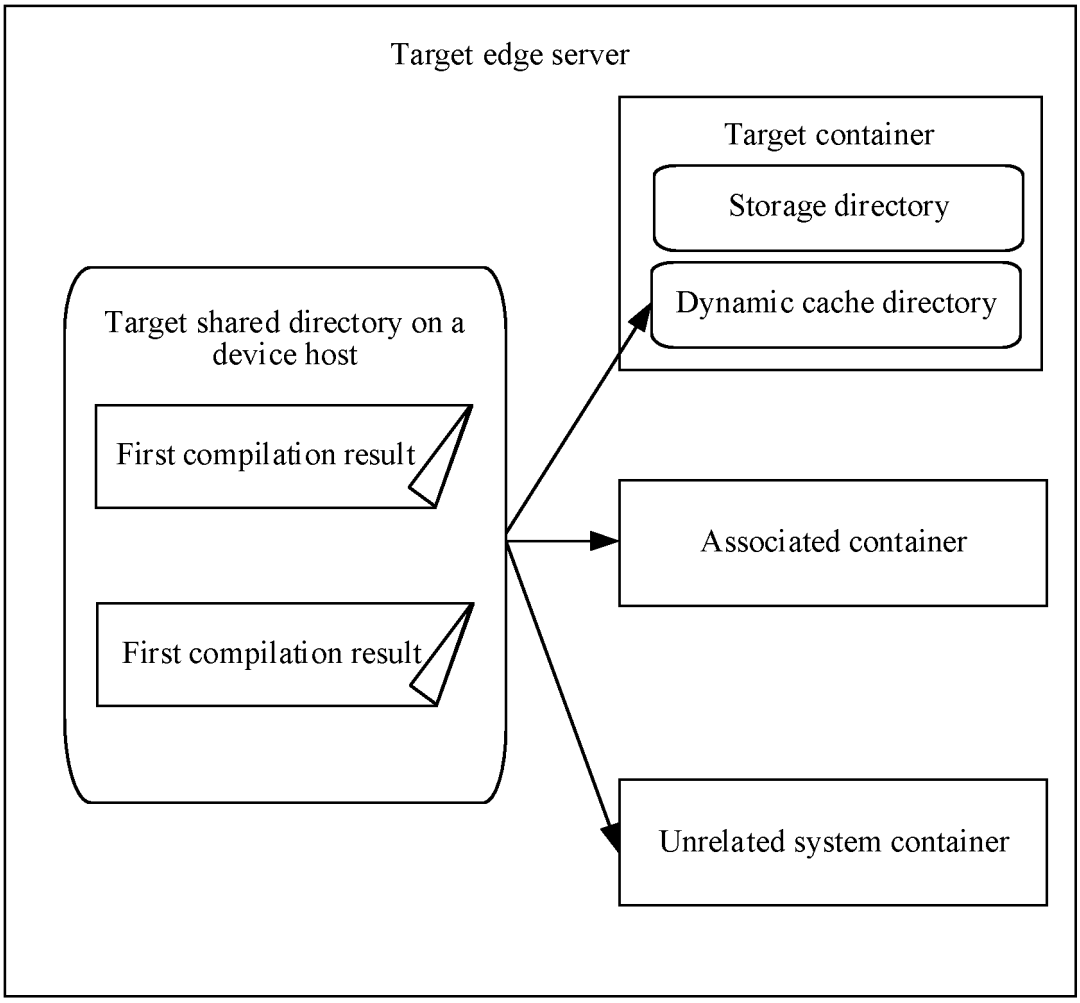
Target edge server
Target container
Storage directory
Dynamic cache directory
Target shared directory on a device host
First compilation result
First compilation result
Associated container
Unrelated system container

FIG. 5a

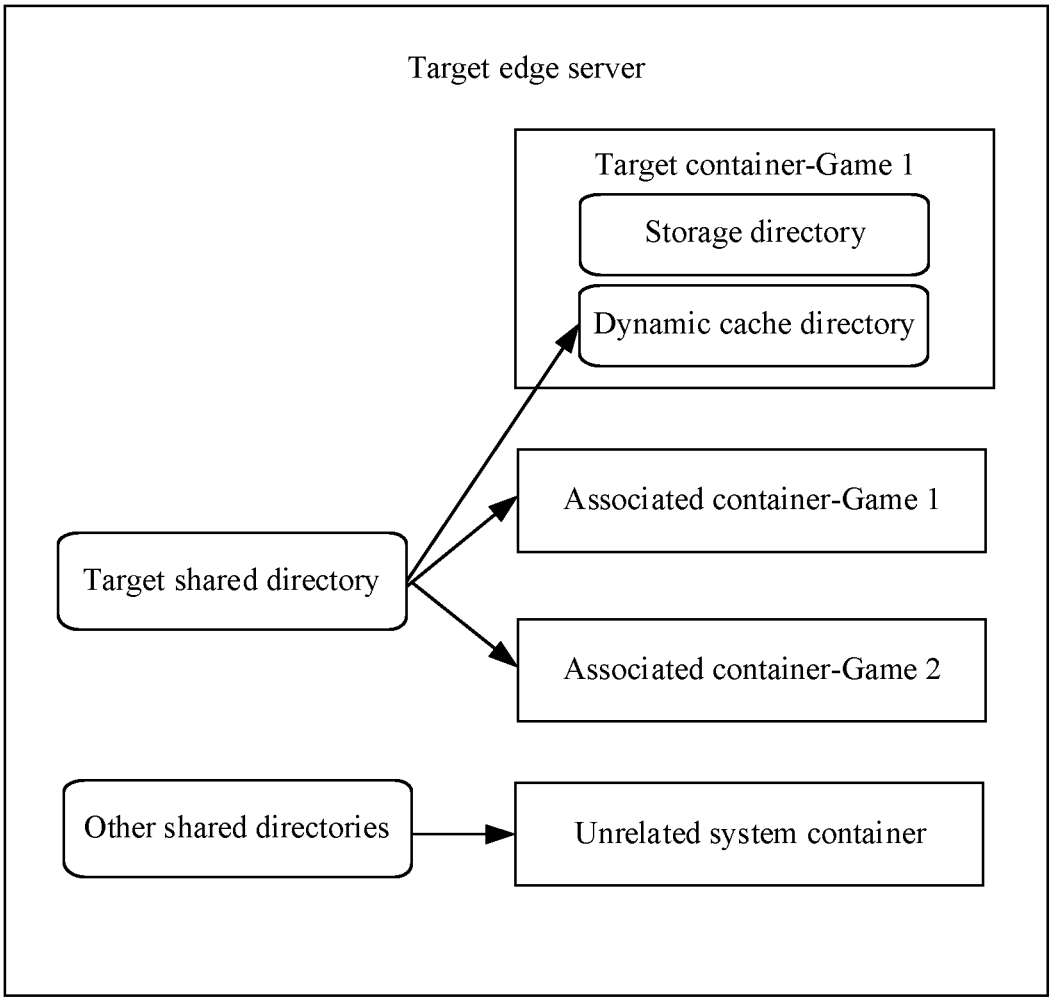
Target edge server
Target container-Game 1
Storage directory
Dynamic cache directory
Associated container-Game 1
Target shared directory
Associated container-Game 2
Other shared directories
Unrelated system container

FIG. 5b

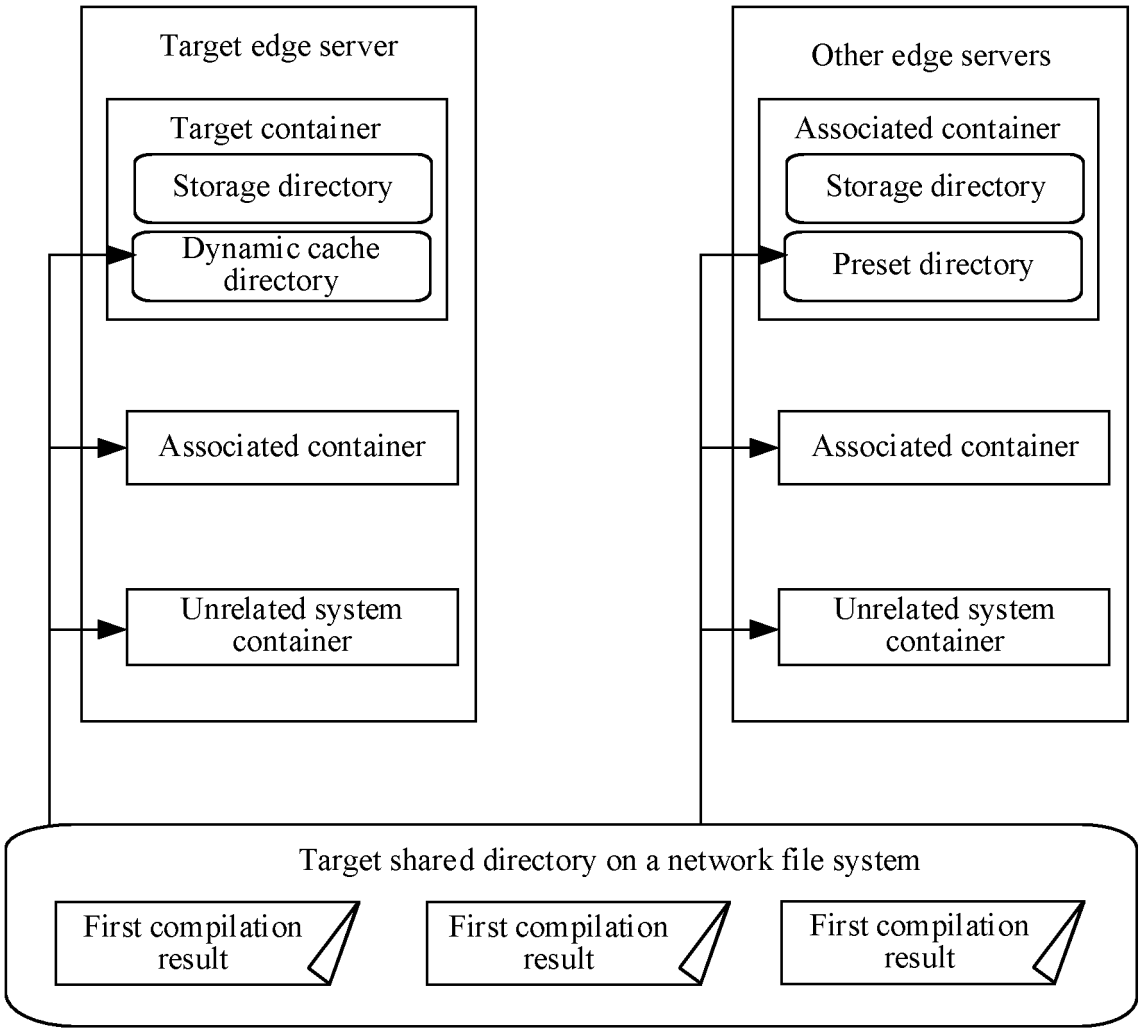
Target edge server
Target container
Storage directory
Dynamic cache directory
Associated container
Unrelated system container
Other edge servers
Associated container
Storage directory
Preset directory
Associated container
Unrelated system container
Target shared directory on a network file system
First compilation result
First compilation result
First compilation result

FIG. 5c

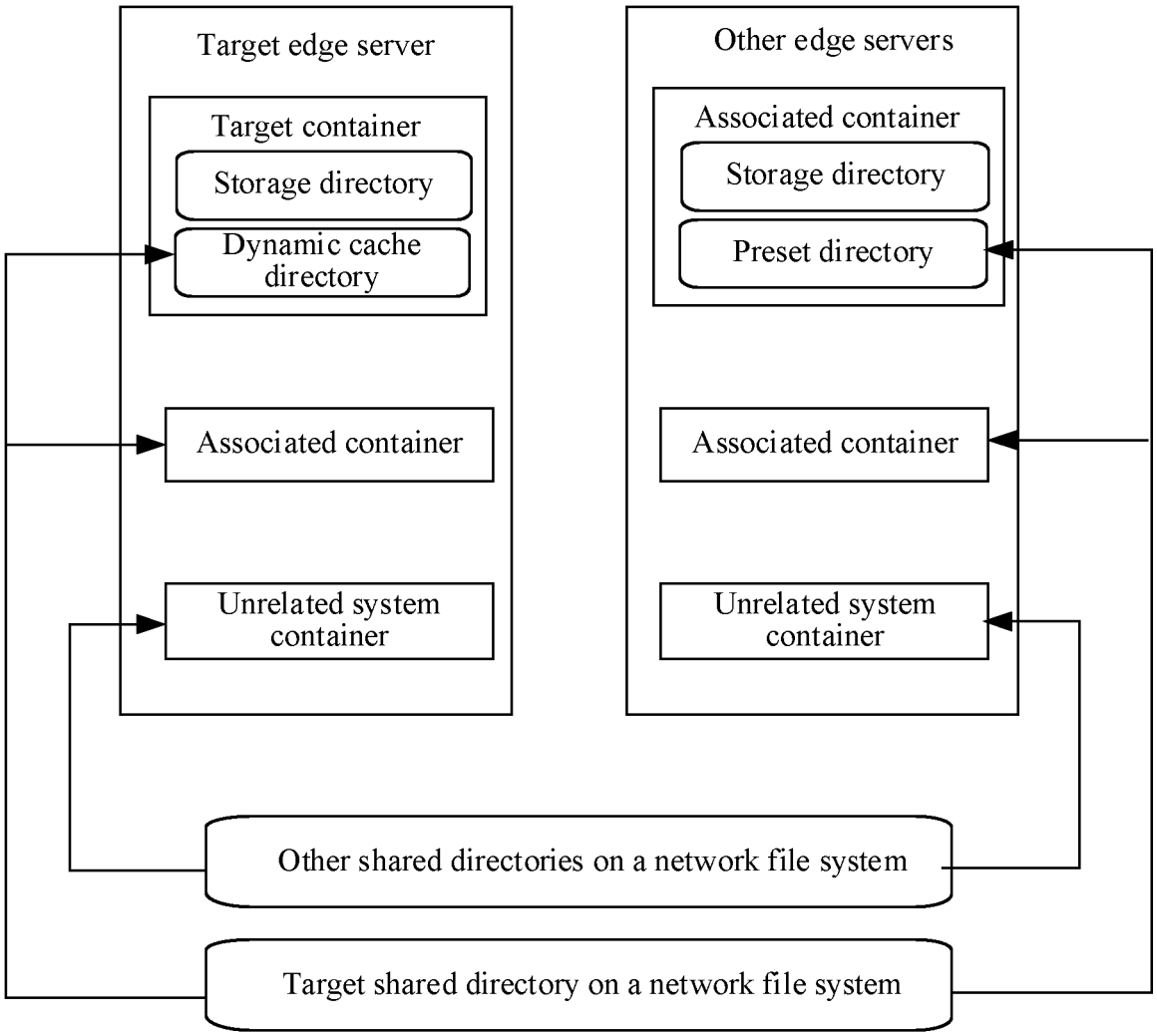
Target edge server
Target container
Storage directory
Dynamic cache directory
Associated container
Unrelated system container
Other edge servers
Associated container
Storage directory
Preset directory
Associated container
Unrelated system container
Other shared directories on a network file system
Target shared directory on a network file system

FIG. 5d

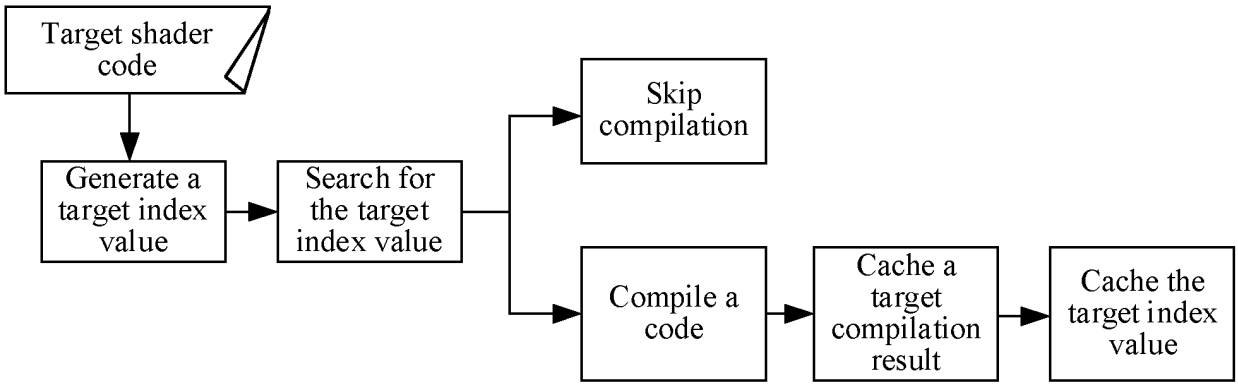
Target shader code
Generate a target index value
Search for the target index value
Skip compilation
Compile a code
Cache a target compilation result
Cache the target index value

FIG. 5e

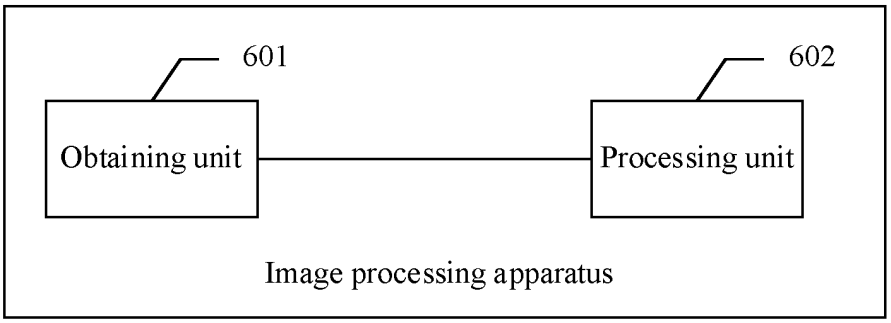
601
602
Obtaining unit
Processing unit
Image processing apparatus

FIG. 6

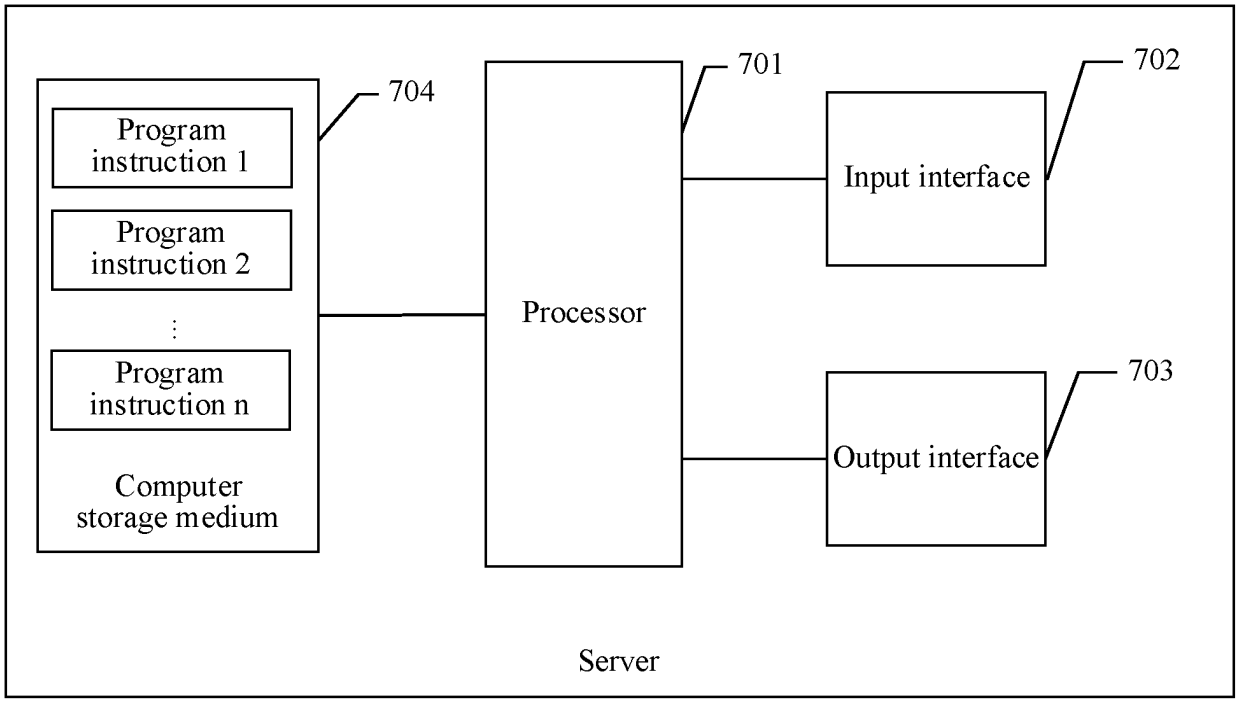
Program instruction 1
Program instruction 2
Program instruction n
Computer storage medium
704
Processor
701
Input interface
702
Output interface
703
Server

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2021/100026

A. CLASSIFICATION OF SUBJECT MATTER

G06F 8/41(2018.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI; EPODOC; CNPAT; CNKI; IEEE: 图像, 处理, 容器, 运行, 游戏, 操作, 事件, 获取, 对应, 目标, 着色, 代码, 资源, 编译, 镜像, 共享, 目录, 反馈, container, corresponding, event, processing, image, obtaining, target, color, code, operation, game

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	CN 111736850 A (TENCENT TECHNOLOGY SHENZHEN CO., LTD.) 02 October 2020 (2020-10-02) claims 1-12, description paragraphs [0013]-[0082]	1-20
X	WO 2019147974 A2 (VALVE CORPORATION) 01 August 2019 (2019-08-01) description paragraphs [0014]-[0093]	1-20
A	CN 110933036 A (MIGU INTERACTIVE ENTERTAINMENT CO., LTD. et al.) 27 March 2020 (2020-03-27) entire document	1-20
A	US 2016162272 A1 (TENCENT TECHNOLOGY (SHENZHEN) COMPANY LIMITED) 09 June 2016 (2016-06-09) entire document	1-20

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
02 August 2021	**27 August 2021**
Name and mailing address of the ISA/CN **China National Intellectual Property Administration (ISA/CN)** **No. 6, Xitucheng Road, Jimenqiao, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer Telephone No.

Form PCT/ISA/210 (second sheet) (January 2015)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2021/100026

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 111736850 A	02 October 2020	None	
WO 2019147974 A2	01 August 2019	KR 20200115557 A	07 October 2020
		EP 3735679 A2	11 November 2020
		US 2019232164 A1	01 August 2019
		CN 111615716 A	01 September 2020
CN 110933036 A	27 March 2020	None	
US 2016162272 A1	09 June 2016	WO 2015021860 A1	19 February 2015
		CN 104375849 A	25 February 2015

Form PCT/ISA/210 (patent family annex) (January 2015)

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- CN 202010704005 **[0001]**